(12)

COMPUTER VISION RESEARCH AND ITS APPLICATIONS TO AUTOMATED CARTOGRAPHY

Second and Third Semiannual Technical Reports
Covering the period June 11, 1983 to June 10, 1984

Contract Amount:	$3,654,877
Effective Date:	December 10, 1982
Expiration Date:	September 30, 1985

September 1984

By: Martin A. Fischler, Program Director
Principal Investigator, (415) 859-5106

Artificial Intelligence Center
Computer Science and Technology Division

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

Attention: *Commander Ronald Ohlander, Program Manager*
Information Processing Techniques Office

Contract No. MDA903-83-C-0027
DARPA Order No. 3862 and AMD 8
Program Code No. 3D30, Program Element 61101E
SRI Project 5355

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025-3493
Telephone: (415) 326-6200
Cable: SRI INTL MPK
TWX: 910-373-2046
Telex: 334 486

SRI International

COMPUTER VISION RESERCH AND ITS APPLICATIONS TO AUTOMATED CARTOGRAPHY

Second and Third Semiannual Technical Reports
Covering the period June 11, 1983 to June 10, 1984

Contract Amount:	$3,654,877
Effective Date:	December 10, 1982
Expiration Date:	September 30, 1985

September 1984

By: Martin A. Fischler, Program Director
Principal Investigator, (415) 859-5106

Artificial Intelligence Center
Computer Science and Technology Division

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

Attention: Commander Ronald Ohlander, Program Manager
Information Processing Techniques Office

Contract No. MDA903-83-C-0027
DARPA Order No. 3862 and AMD 8
Program Code No. 3D30, Program Element 61101E
SRI Project 5355

Approved:

Stanley J. Rosenschein, Director
Artificial Intelligence Center

Donald L. Nielson, Acting Director
Computer Science and Technology Division

SRI International®

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: 2nd & 3rd Semiannual Tech. Rpts.	2. GOVT ACCESSION NO.: AD-A146668
	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): Computer Vision Research and Its Applications to Automated Cartography	5. TYPE OF REPORT & PERIOD COVERED: Combined Semiannual Technical 6/11/83 - 12/10/83 & 12/11/83-6/10/84
	6. PERFORMING ORG. REPORT NUMBER: 5355 2nd & 3rd Semiann. Tech.
7. AUTHOR(s): Martin A. Fischler	8. CONTRACT OR GRANT NUMBER(s): MDA903-83-C-0027
9. PERFORMING ORGANIZATION NAME AND ADDRESS: SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Program Code No. 3D30, Program Element 61101E
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209	12. REPORT DATE: September 1984
	13. NUMBER OF PAGES: 130 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): DCASMA, San Francisco, 1250 Bayhill Drive, San Bruno, California 94066	15. SECURITY CLASS (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

image understanding, computer vision, automated cartography, feature extraction, stereo compilation, linear delineation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The SRI Image Understanding program is a broad effort spanning the entire range of machine vision research. Its three major concerns are: (1) to develop an understanding of the physics and mathematics of the vision process, (2) to develop a knowledge-based framework for integrating and reasoning about sensed (imaged) data, and (3) to develop a machine-based environment for effective experimentation, demonstration, and evaluation of our theoretical results, as well as providing a vehicle for technology transfer. This report describes recent progress in all three areas. In

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT (continued)

particular, we have shown that fractal functions are an effective tool for representing natural shapes and provide a good basis for recovering 3-D shape from the shading and texture in a single image. For scenes containing man-made objects, we have found ways of using straight edges to recover the 3-D orientation of surfaces from a single view, and reason about the shape of an object from partial information in multiple views. We have built a new and powerful LISP-machine-based environment for use in image understanding research, and are putting together high-performance systems for stereo compilation, feature extraction, and linear delineation.

TABLE OF CONTENTS

PREFACE

This report combines the semiannual technical reports for the periods June 11, 1983 through December 10, 1983 and December 11, 1983 through June 10, 1984.

1. INTRODUCTION

The goal of this research program is to obtain solutions to fundamental problems in computer vision, particularly to such problems as stereo compilation, feature extraction, linear delineation, and general scene modeling that are relevant to the development of an automated capability for interpreting aerial imagery and the production of cartographic products.

To achieve this goal, we are engaged in investigations of such basic issues as image matching, partitioning, representation, and physical modeling (shape from shading, texture, and optic flow; material identification; recovery of imaging and illumination parameters such as "vanishing points," "camera parameters," illumination source location; edge classification; etc.). However, it is obvious that high-level, high-performance vision requires the use of both intelligence and stored knowledge (to provide an integrative framework), as well as an understanding of the physics and mathematics of the imaging process (to provide the basic information needed for a reasoned interpretation of the sensed data). Thus, a significant portion of our work is devoted to developing new approaches to the problem of "knowledge-based vision." Finally, vision research cannot proceed without a means for effective implementation, demonstration, and experimental verification of theoretical concepts; we have developed an environment in which some of the newest and most effective computing instruments can be employed for these purposes.

2. KNOWLEDGE BASED VISION: the Construction of an Expert System Control Structure for Stereo Compilation and Feature Extraction.

Our intent in this effort is to develop a system framework for allowing higher level knowledge to guide and integrate the detailed interpretation of imaged data by autonomous scene analysis techniques. Such an approach allows symbolic knowledge, provided by higher-level knowledge sources, to automatically control the selection of appropriate algorithms, adjust their parameters, and apply them in the relevant portions of the image. More significantly, we are attempting to provide an efficient means for supplying and using qualitative knowledge about the semantic and physical structure of a scene so that the machine-produced interpretation, constrained by this knowledge, will be consistent with what is generally true of the overall scene structure, rather than just a good fit to locally applied models.

An important component of our approach is to design a means for a human operator to simply and effectively provide the machine with a qualitative scene description, in the form of a semantically labeled 3-D "sketch." This capability for effective communication between a human and the machine about the three-dimensional world requires both appropriate graphics tools and an ability on the part of the machine for both spatial reasoning and some semantic "understanding." The importance of this work derives from the fact that a major difficulty in automating the image-interpretation process is the inability of current computer systems to deduce, from the visible image content, the general context of the scene (e.g., urban or rural; season of the year; what happened immediately before, and what will

happen immediately after, the image was viewed by the sensor) – the knowledge base and reasoning required for such an ability is well beyond what the state of our art can hope to accomplish over (at least) the next 5 years. Thus, our work is intended to provide an interim means by which a human can supply, a task-oriented program, the high level overview it needs for its analysis, but cannot acquire by itself.

Two of our major integrative efforts are directly concerned with the knowledge-based vision problem:

One effort, integrating our work in stereo compilation and physical modeling, is the construction of a rule-based system with a library of processes and activities, which can be invoked to carry out specific goals in the domain of cartographic analysis and stereo reconstruction. This work is based on results described below, but the integrative framework is still being developed and will not be described in this report.

The second effort, described in a following section on feature extraction, is a restricted version of the concept discussed above (it employes contextual and semantic knowledge, but does not address the issues of qualitative reasoning nor 3-D spatial understanding).

3. DEVELOPMENT OF METHODS FOR MODELING AND USING PHYSICAL CONSTRAINTS IN IMAGE INTERPRETATION.

Our goal in this work is to develop methods that will first allow us to produce a sketch of the physical nature of a scene and the illumination and imaging conditions, and then permit us to use this physical sketch to guide and constrain the more detailed descriptive processes – such as precise stereo mapping.

Our approach is to develop:

- models of the relationship between physical objects in the scene and the intensity patterns they produce in an image (e.g., models that allow us to classify intensity edges in an image as either shadow, or occlusion, or surface intersection, or material boundaries in the scene),
- models of the geometric constraints induced by the projective imaging process (e.g., models that allow us to determine the location and orientation of the camera that acquired the image, location of the vanishing points induced by the interaction between scene and camera, location of a ground plane, etc.), and
- models of the illumination and intensity transformations caused by the atmosphere, light reflecting from scene surfaces, and the film and digitization processes that result in the computer representation of the image.

These models, when instantiated for a given scene, provide us with the desired "physical" sketch. We are assembling a "constraint-based stereo system" that can use this physical sketch to resolve the ambiguities that defeat conventional approaches to stereo modeling of scenes (e.g., urban scenes or scenes of cultural sites) for which the images are widely separated in either space or time, or for which there are large featureless areas, or a significant number of occlusions.

A summary of some of our current work in the area of modeling and using physical constraints is presented below.

Rectilinear Forms. Images of cultural scenes, such as building complexes, typically contain a significant amount of linear structure. We have developed an effective computational technique for recovering 3-D interpretations from a single 2-D image in many such cases. It works by finding a basis for a vector space suitable for quantifying spatial relations, while satisfying constraints imposed by linear features in the image. Given three image lines that are assumed to be perspective projections of mutually orthogonal scene features, the method backprojects the lines into three-dimensional scene space, generating (potentially) all possible combinations of line orientations. It selects the combination that is "most orthogonal" by maximizing the triple product of three unit basis vectors, using the method of steepest descent. In general, two solutions are found, and the correct one can be chosen by relating the solutions to knowledge of the vertical direction. A more complete description of this work is presented in Barnard [1984a] (Appendix A).

Inductive Approach. The technique discussed above has led us to investigate a new class of computational methods for the interpretation of single images. These methods constitute an inductive approach because they explicitly recognize that the available data (the image) are insufficient to make a well-founded logical interpretation; that is, many interpretations are possible, but only one is preferred. Specific prior models cannot account for the general power of such perception in the case of a human observer (although prior models are certainly used when available). To be truly general purpose, machine vision must be able to mimic this amazing human ability. The inductive approach selects interpretations that are "simplest" in some sense. While it does not preclude the use of specific prior models, it emphasizes the use of abstract generic models, such as parametric curves and surfaces. One measure of simplicity we have considered is based on information-theoretic considerations. This work will be described in a report by Barnard [1984b].

Optic Flow. In the optic flow paradigm, a moving observer is normally able to interpret a time sequence of images as an implicit description of a static scene. In principle, the images can be matched point-by-point and the motion of the observer can be deduced by exploiting the constraint that the scene is fixed. In practice, this is exceedingly difficult to achieve, both because point features are rare and because the methods are very sensitive to small matching errors.

We have developed an alternative optic-flow method that exploits the often available information about the rotation of the observer. Knowing the observer's rotation greatly simplifies the problem of matching successive images, but, since all the useful information that can be derived from the sequence is due to the translation of the observer, it does not significantly sacrifice generality. The major advantage to translation-only optic flow is that curvilinear image features can be matched by exploiting a constraint that is essentially the same as the epipolar constraint in stereo interpretation. This work is still in progress.

Spatial Reasoning from Line Drawings of Polyhedra. Construction of a three-dimensional "sketch" is one task faced by the user of an interactive image understanding expert system. An urban scene typically contains buildings and other objects that can be modeled as planer-faced polyhedra. An effective way for the user to create 3-D sketches from multiple views of such objects has been devised.

The system requires two or more line drawings of a polyhedral scene from arbitrary

vantage points. These line drawings may be obtained from a freehand sketch, by tracing the edges in several photos, or from the output of an automatic edge detector. A "wireframe" model of the objects is obtained by back-projecting the line drawings. Labels of solid or vacant space are then assigned to all spatial regions defined by the wireframe using an iterative constraint propagation algorithm. The result is a data structure that captures the volumetric structure of the objects depicted, which can then be used to support hidden-line elimination and other volumetric operations upon the object. This work is described in Strat [1984a] (Appendix B).

Determining The Imaging Geometry from a Camera Transformation Matrix. Many scene analysis algorithms require knowledge of the geometry of the image formation process as a prerequisite to their application. When the imaging situation can be controlled or measured directly, the needed parameters can be determined; however, in the case of uncalibrated images, or photographs whose history is unknown, the necessary parameters are not available. In these cases, an alternate method of inferring the imaging situation from the correspondence between a small set of image and object points is required.

One approach has been to compute the imaging geometry directly from the constraints provided by the known data points. Partial information such as the camera's focal length or the location of the piercing point in the image can be used to reduce the number of data points needed. A second approach consists of two steps. First, the known data points are used to compute a 4x4 homogeneous coordinate transformation matrix that captures the entire transformation from object point to image point. An established technique for this computation involves the least squares solution of a set of simultaneous linear equations from six or more known correspondences. The goal of the second step is to derive the various parameters of the image formation process from the transformation matrix. This problem can be posed as a system of nonlinear equations whose solution had required iterative methods. Recently, Ganapathy [1984] published the first noniterative solution.

Research performed at SRI has also produced a noniterative solution (Strat [1984b] [Appendix C]). By reasoning about the geometric constraints inherent in a camera transformation matrix, a simple, easily understood method of determining the various parameters is obtained. Through a series of geometric constructions, the camera's location and orientation, along with the piercing point and the relations between the focal length and scale factors, can be determined. The method relies purely on spatial reasoning about geometric constraints and does not involve an intuitively opaque matrix decomposition. Furthermore, its sensitivity to errors can be studied geometrically, allowing a clear understanding of the conditions that lead to inaccurate decompositions. The technique has been successfully applied to both synthetic imaging situations and real photographs.

4. STEREO COMPILATION: IMAGE MATCHING AND INTERPOLATION

We are implementing a state-of-the-art stereo system that produces dense range images given pairs of intensity images. We plan to use it both as a framework for our stereo research, and as the base component of an expert system concerned with 3-D compilation.

There are five components of this stereo system: a rectifier, a sparse matcher, a dense matcher, an interpolater, and a projective display module. The rectifier accepts estimates of the parameters and distortions associated with the imaging process, the photographic process, and the digitization. These parameters are used to map digitized image coordinates onto an ideal image plane. The sparse matcher performs two-dimensional searches to find several matching points in the two images, which it uses to compute a relative camera model. The dense matcher tries to match as many points as possible in the two images. It uses the relative camera model to constrain the searches to one dimension, along epipolar lines. The interpolater computes a grid of range values by interpolating between the matches found by either the sparse or the dense matcher. The projective display module allows interactive examination of the computed 3-D model by generating 2-D projective views of the model from arbitrarily selected locations in space.

The current system, which runs on the VAX/11-780 in C, is described in Hannah [1984] (Appendix D). At present, the system produces relatively sparse 3-D information, even in its dense matching mode. Often 3-D data are required that are more closely spaced than can be provided by the stereo matching process. Further, there may be areas of the images that cannot be matched due to noise, insufficient information, and occlusions; this will produce holes in the dense matched data that must be filled in. In either case, interpolation is necessary to provide 3-D data between matched points.

Interpolation. We are currently exploring two different schemes for interpolation. One is a global approach, in which all of the 3-D information available is used to find the interpolated value for a given point. (This approach is described in Smith [1984] [Appendix E].) The second approach is a local one, which only uses the data in the neighborhood of the point to be interpolated. The global approach produces a functional description that can be differentiated analytically to determine slope and other surface attributes; the local approach is most useful in the context of verifying the plausibility of the matches by comparing the data from the stereo images after projection onto this surface. The local approach is being used in the context of a hierarchical matching scheme described below.

Matching. In a parallel research effort employing our Lisp Machines, we are exploring a hierarchical technique for developing a regular, dense grid of matched points. This technique does appropriate warping of the images between each level of the hierarchy, to account for differences in perspective between the two images as predicted by the model. As a part of this effort, local interpolation techniques have been developed to fill in holes in the model before proceeding from level to level.

The Lisp Machine implementation includes a sophisticated terrain display package, which permits the user to interactively designate a flight path through the 3-dimensional model derived from a pair of images; the system then creates a "movie" (a sequence of either monocular or stereo views) of the terrain as the user "flies" along the path above the terrain. This package is useful not only for assessing the quality of the derived model, but also for tasks in which a prediction of the appearance of the scene from arbitrarily specified points of view is desired, as when an observer is moving through mapped terrain. This work is described in Quam [1984] (Appendix F).

Evaluation. We now have available, on our VAX (Testbed) and Lisp Machines, some of the most advanced stereo matching systems developed by the IU community. As a part of our stereo research effort, we plan to run several calibrated data sets through these systems to determine the relative strengths and weaknesses of the various methods, including area correlation, hierarchical warped matching, edge matching, and edge/intensity matching.

5. THE REPRESENTATION OF NATURAL SCENES

Our current research in this area addresses two related problems: (1) representing natural shapes such as mountains, vegetation, and clouds, and (2) computing such descriptions from image data. The first step towards solving these problems is to obtain a model of natural surface shapes.

A model of natural surfaces is extremely important because we face problems that seem impossible to address with standard descriptive computer vision techniques. How, for instance, should we describe the shape of leaves on a tree? Or grass? Or clouds? When we attempt to describe such common, natural shapes using standard representations, the result is an unrealistically complicated model of something that, viewed introspectively, seems very simple. Furthermore, how can we extract 3-D information from the image of a textured surface when we have no models that describe natural surfaces and how they evidence themselves in the image? The lack of such a 3-D model has restricted image texture descriptions to being ad hoc statistical measures of the image intensity surface.

Fractal functions, a novel class of naturally arising functions, are a good choice for modeling natural surfaces, because many basic physical processes (e.g., erosion and aggregation) produce a fractal surface shape and because fractals are widely used as a graphics tool for generating natural-looking shapes. Additionally, in a survey of natural imagery, we found that a fractal model of imaged 3-D surfaces furnishes an accurate description of both textured and shaded image regions, thus providing validation of this physics-derived model for both image texture and shading.

Progress relevant to computing 3-D information from imaged data has been achieved by use of the fractal model. A test has been derived to determine whether or not the fractal model is valid for a particular set of image data, an empirical method for computing surface roughness from image data has been developed, and substantial progress has been made in the areas of shape-from-texture and texture segmentation. Characterization of image texture by means of a fractal surface model has also shed considerable light on the physical basis for several of the texture-partitioning techniques currently in use and made it possible to describe image texture in a manner that is stable over transformations of scale and linear transforms of intensity.

The computation of a 3-D fractal-based representation from actual image data has been demonstrated. This work has shown the potential of a fractal-based representation for efficiently computing good 3-D representations for a variety of natural shapes, including such seemingly difficult cases as mountains, vegetation, and clouds.

This research is expected to contribute to the development of (1) a computational theory of vision applicable to natural surface shapes, (2) compact representations of shape useful

for natural surfaces, and (3) real-time regeneration and display of natural scenes. We also anticipate adding significantly to our understanding of the way humans perceive natural scenes.

Details of this work can be found in Pentland [1983 and 1984] (1984, Appendix G).

6. FEATURE EXTRACTION: SCENE PARTITIONING, AND LABELING

Our efforts in image partitioning and labeling have advanced along two fronts: we have developed a goal-directed texture-based segmentation algorithm and have studied knowledge-based control concepts required to integrate this with other image feature-extraction techniques.

The SLICE goal-directed segmentation system combines knowledge of target textures or signatures with knowledge of background textures by using histogram-similarity transforms. Regions of high similarity to a target texture and of low similarity to any negative texture examples are found. This use of semantic knowledge during the segmentation process improves segmenter performance and focuses segmentation activity on material types of greatest interest. (The system can also be used for goal-independent texture segmentation by omitting the similarity-transform computations.) Development of this segmentation technique is essentially complete; all that remains is to integrate it into the more general feature-extraction system described below. Performance of the SLICE segmentation algorithm is documented in Laws [1984] (Appendix H).

The KNIFE (knowledge-based interactive feature-extraction) system is intended to solve problems in image segmentation, feature extraction, material identification, and feature classification. (Image segmentation and feature extraction partition an image into meaningful units; material identification and feature classification label those units.) Experience has shown that these tasks cannot be carried out adequately in isolation. Image segmentation, for instance, cannot produce a meaningful partitioning unless it is guided by semantic criteria from material identification and feature classification.

The KNIFE feature-extraction system will combine a data base of recognition rules (using shape, texture, and context) with recursive segmentation and other techniques to find and label scene features. Initially selected image regions, based on image brightness and texture, are resegmented and refined to locate recognizable objects (e.g., roads, fields, and buildings). The control process assigns initial labels for each region, and then recursively analyzes those regions that might contain useful substructure. The choice of regions to split or merge is influenced by analysis goals rather than solely by statistical properties of the image data. The segmentation and interpretation will thus proceed at unequal rates or to different depths in separate scene regions, with differing types of knowledge applied at successive stages in the analysis. Objects detected by other means (user interaction or direct object recognition) may override the normal interpretation cycle.

We are concentrating our development efforts on goal-directed recursive segmentation and on related display, query, and editing tools. Among these tools are display of input images and segmentation maps; readout of region descriptions and relationships; and com-

mands for interactively designating, splitting, merging, and classifying regions.

The control process is a production system that looks for applicable rules in the rule base. Such rules will be placed on a prioritized queue of tasks to be performed. When executed, they may query the user, invoke image analysis subsystems, or affect the behavior of the control process itself.

Besides the rule base and the input or derived imagery, the system will have two principal data structures. These are the sketch data base, and the prototype data base.

The sketch data base serves as the system blackboard, storing all the information relevant to the current image. The prototype data base will be a semantic network with nodes storing object properties and pointers to image examples.

The system is being developed on the VAX-based SRI Image Understanding (IU) Testbed. The basis for the system's data analysis capabilities will be the body of software currently accumulated in the testbed and other programs now being developed, such as the SLICE goal-directed segmentation system discussed above.

7. LINEAR DELINEATION AND PARTITIONING

A basic problem in machine-vision research is how to produce a line sketch that adequately captures the semantic information present in an image. (For example, maps are stylized line sketches that depict restricted types of scene information.) Before we can hope to attack the problem of semantic interpretation, we must solve some open problems concerned with direct perception of line-like structure in an image, and with decomposing complex networks of line-like structures into their primitive (coherent) components. Both of these problems have important practical as well as theoretical implications.

For example, the roads, rivers, and rail-lines in aerial images have a line-like appearance. Methods for detecting such structures must be general enough to deal with the wide variety of shapes they can assume in an image as they traverse natural terrain.

Most approaches to object recognition depend on using the information encoded in the geometric shape of the contours of the objects. When objects occlude or touch one another, decomposition of the merged contours is a critical step in interpretation.

We have made significant progress in both the delineation and the partitioning problems. Our work in delineation (Fischler and Wolf [1983]) is based on the discovery of a new perceptual primitive that is highly effective in locating line-like (as opposed to edge-like) structure.

One approach to decomposing linear structures into coherent components (Fischler and Bolles [1983]) is based on the concept that perception is an explanatory process – acceptable precepts must be associated with explanations that are believable: They must be complete (i.e., they explain all the data), simple (i.e., both concise and of limited complexity), and stable (i.e., they must not change under small perturbations of either the imaging conditions or the decision algorithm parameters).

A second approach to the partitioning problem, which also addresses the problem of qualitative matching of linear structures (Smith and Wolf [1984] [Appendix I]), focuses on the concept of simplicity as the basis for making perceptual decisions. Given a set of primitives

as the basis for description, each possible description of a set of data is evaluated as to how accurately it describes the data and how "long" a description is required (a natural conversion from accuracy to descriptive length is provided). The shortest description is chosen as being correct.

These new delineation and partitioning algorithms have produced excellent results in experimental tests on real data. Our continuing work in this area focuses on theoretical, as well as performance, issues.

8. COMPUTING ENVIRONMENT FOR IU RESEARCH

Previous reports (e.g., Hanson and Fischler [1981]) describe the VAX 11/780 testbed environment we created for evaluation, demonstration, and transfer of IU technology. A significant recent addition to this system is based on the Symbolics 3600 LISP machine. Documentation of this new system is still incomplete, but as noted in section four of this report (Stereo Compilation: matching), applications recently considered beyond the state-of-the-art on comparably priced hardware, have already been programmed and demonstrated (Quam [1984]).

Acknowledgement

The following researchers have contributed to the work described in this report: H. Baker, S. Barnard, R.C. Bolles, M.A. Fischler, M.J. Hannah, A.J. Hanson, D.L. Kashtan, K. Laws, O. Firschein, A.P. Pentland, L.H. Quam, G.B. Smith, T. Strat, and H.C. Wolf.

References

S.T. Barnard, *Choosing a Basis for Perceptual Space, Proc. IEEE Workshop on Computer Vision: Representation and Control,* Annapolis, Maryland, (April 1984) (to appear in *Computer Vision, Graphics, and Image Processing*). Also Appendix A of this report.

S.T. Barnard, "An Inductive Approach to Figural Perception," Technical Note (in preparation), Artificial Intelligence Center, SRI International, Menlo Park, California (1984).

M.A. Fischler and H.C. Wolf, "Machine Perception of Linear Structure," *Proc. 8th Int. Jnt. Conf. on Artificial Intelligence,* Karlsruhe, West Germany, pp. 1010–1013 (8–12 August 1983).

M.A. Fischler and R.C. Bolles, "Perceptual Organization and the Curve Partitioning Problem," *Proc. 8th Int. Jnt. Conf. on Artificial Intelligence,* Karlsruhe, West Germany, pp. 1014–1018 (8–12 August 1983).

S. Ganapathy, "Decomposition of Transformation Matrices for Robot Vision," *IEEE,* pp. 130-139 (1984).

M.J. Hannah, "Description of SRI's Baseline Stereo System," Technical Note (in preparation), Artificial Intelligence Center, SRI International, Menlo Park, California (1984). Also

Appendix D of this report.

A.J. Hanson and M.A. Fischler, "The DARPA/DMA Image Understanding Testbed" *Proc. DARPA Image Understanding Workshop,* Palo Alto, California, (15–16 September 1982).

K.I. Laws, *Goal-Directed Textured-Image Segmentation,* Technical Note 334, Artificial Intelligence Center, SRI International, Menlo Park, California (September 1984). Also Appendix H of this report.

A.P. Pentland, "Fractal-Based Description," *Proc. 8th Int. Jnt. Conf. on Artificial Intelligence,* Karlsruhe, West Germany, pp. 973–981 (8–12 August 1983).

A.P. Pentland, "Shading Into Texture," *Proc. Nat. Conf. on Artificial Intelligence,* Austin, Texas, pp. 269–273 (6–10 August 1984). Also Appendix G of this report.

L.H. Quam, "Hierarchical Warp Stereo," *Proc. DARPA Image Understanding Workshop,* New Orleans, Louisiana, (3–4 October 1984). Also Appendix F of this report.

G.B. Smith, *A Fast Surface Interpolation Technique,* Technical Note 333, Artificial Intelligence Center, SRI International, Menlo Park, California (August 1984). Also Appendix E of this report.

G.B. Smith and H.C. Wolf, *Image-to-Image Correspondence: Linear-Structure Matching,* Technical Note 331, Artificial Intelligence Center, SRI International, Menlo Park, California (July 1984). Also Appendix I of this report.

T.M. Strat, "Spatial Reasoning from Line Drawings of Polyhedra," *Proc. IEEE Workshop on Computer Vision: Representation and Control,* Annapolis, Maryland (April 1984). Also Appendix B of this report.

T.M. Strat, "Recovering the Camera Parameters from a Transformation Matrix," *Proc. DARPA Image Understanding Workshop,* New Orleans, Louisiana (October 1984). Also Appendix C of this report.

APPENDIX A

Choosing a Basis for Perceptual Space

By: Stephen T. Barnard

CHOOSING A BASIS FOR PERCEPTUAL SPACE

By: Stephen T. Barnard, Senior Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

Abstract

If it is possible to interpret an image as a projection of rectangular forms, there is a strong tendency for people to do so. In effect, a mathematical basis for a vector space appropriate to the world, rather than to the image, is selected. A computational solution to this problem is presented. It works by backprojecting image features into three-dimensional space, thereby generating (potentially) all possible interpretations, and by selecting those which are maximally orthogonal. In general, two solutions that correspond to perceptual reversals are found. The problem of choosing one of these is related to the knowledge of verticality. A measure of consistency of image features with a hypothetical solution is defined. In conclusion, the model supports an information-theortic interpretation of the Gestalt view of perception.

1. Introduction

Why do we see the pattern of lines in Figure 1 as a right-angled corner? First, we must recognize that this is an **illusion**. (Let's call it the "right-angle illusion.") There is no strict, logical reason to interpret this figure in such a way: there are infinitely many three-dimensional spatial configurations of line segments that could have "explained" it. Nevertheless, we do see it in a special way — thus, we experience an illusion. Is it possible to understand this from a computational point-of-view?

The right-angle illusion does not depend on the three lines meeting at a common vertex. The pattern in Figure 2 evokes a comparable impression and it has no common vertex. The illusion is strengthened by rotating the pattern so that one line can be seen as vertical and the others as horizontal. This can be checked by rotating Figure 2 ninety degrees. Does the illusion still seem as vivid? On the other hand, more complex patterns, such as Figure 3, do not necessarily lead to right-angled interpretations, but, in these cases, the viewer is given additional constraining information beyond three line segments. If three line segments form two very acute angles, such as in Figure 4, they will not be seen as right-angled, but then no such interpretation is geometrically possible.

In summary, it seems that, in the absence of additional information, three non-colinear line segments will be seen as perpendicular lines in space, if such an interpretation is possible. There is strong experimental evidence for this hypothesis. Attneave and Frost [1] found that the perceived orientations of lines were highly predictable from hypothetical orientations implied by right-angled interpretations. Perkins [2] tested the ability to discriminate between right-angled and non-right-angled forms. He found that when an image could be explained by a right-angled interpretation it would almost always be perceived in that way.

At first, one might think the right-angle illusions too sparse to be meaningful. They contain so little information. Could they ever compare to real visual experience, with its abundance of data? Consider the familiar Ames-room illusion [3] (Figure 5). A weird, trapezoidal room is contrived to look normal (i.e., rectangular) from a particular point-of-view. Objects in the room are seen incorrectly: a man on one side of the room appears to be a midget, while another on the opposite side appears to be a giant. The Ames room illusion and the right-angle illusion have one critical point in common: in both cases, rectilinear perceptions are constructed from too little evidence. In the Ames room, furthermore, the effect is so strong that it dominates other important information for depth perception (such as size constancy). In an effort to make the distorted room look normal, our perception creates an incorrect geometrical interpretation that implies incorrect metric relations between objects. In effect, we perceive the *space* in which the room and the objects exist, rather than perceiving the room and the objects in isolation.

In this sense, the right-angle illusion is simply a minimal case of the Ames room.

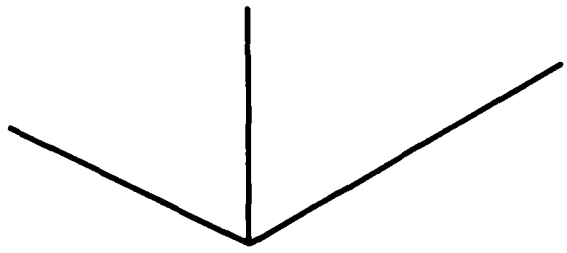


Figure 1: A Right-Angle Illusion

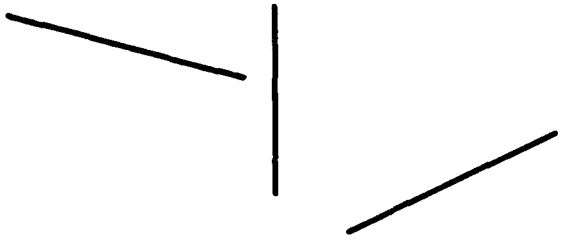


Figure 2: A Common Vertex is not Required

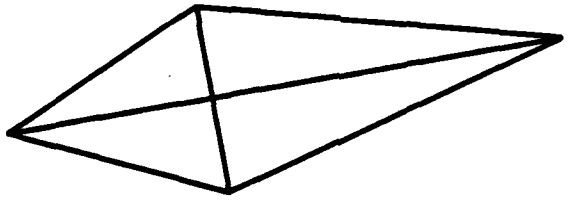


Figure 3: A Tetrahedron

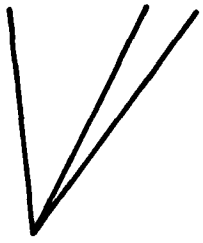


Figure 4: A Pattern that Does Not Admit a Right-Angled Interpretation

The difference between the two is that the Ames room contains abundant information for a rectilinear interpretation, whereas a pattern of three line segments contains the bare minimum. Any three mutually orthogonal lines from the Ames

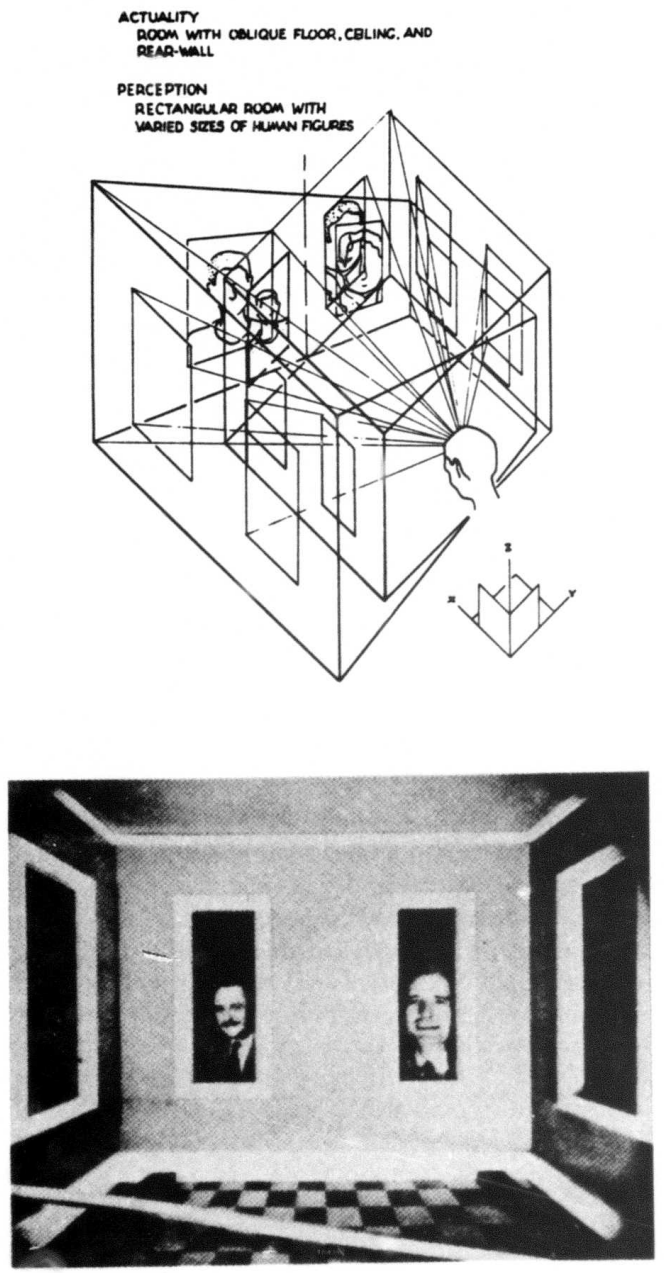


Figure 5: The Ames Room Illusion

room will produce an acceptable right-angle illusion, and, furthemore, all these illusions will be consistent in the sense of aligning with the natural coordinate system of the (imagined) room.

There are two reasons to be interested in this problem. First, a solution might suggest fundamental principles of perception. Human perception is an odd, complex, but remarkably consistent and efficient process. It "reasons" from incomplete evidence and almost never makes a serious error. Understanding this peculiar (and awesome) ability is the central task of vision research [4]. A second, pragmatic reason is that these patterns are quite common in images of natural scenes (see Figure 6). An algorithm that could make sense of them could contribute a basic capability to a larger machine vision system.

An early attack on a similar problem was directed at the so-called "blocks world." (See Mackworth [5] for a good summary of this line of research.) In a paper that pioneered the field of scene analysis, Roberts described a system for recognizing a small collection of simple, generic polyhedral shapes [6]. Whereas Roberts' methods produced complete metric descriptions of scenes, the blocks-world work that followed was aimed at segmentation and qualitative description. The methods were fundamentally syntactic and viewed the problem of blocks-world interpretation as a matter of parsing line drawings into allowable configurations of line and vertex types. The simplification of orthographic projection was introduced, and the effects of perspective were considered irrelevant. To the extent that metric constraints were used [7], [8], they were relatively weak and did not generalize in a straightforward way to perspective.

The approach presented here is quite different. A right-angle illusion, or a more complex image of an Ames room, or a blocks-world scene, or a natural scene such as Figure 6, imply certain interpretations *for geometrical reasons alone*. Specifically, intepretations that are in some sense "orthogonal" are preferred. A method for finding such interpretations for right-angle illusions will be presented. The approach is to seek a three-dimensional description that simultaneously accounts for the two-dimensional figure and the three-dimensional phenomenal perception. In contrast to the blocks-world results, the method is as easily stated for perspective as for orthography, and produces quantitative answers. It has a simple mathematical representation and computer implementation.

Figure 6: A Real Scene with Right-Angle Configurations

2. The Computational Model

In this section a formal mathematical model will be presented as a computational explanation of the right-angle illusion. The model consists of a method for constructing interpretations that are orthogonal and that are in the form of triplets of unit vectors. In essence, the interpretation constructs an alternative **basis** for the perceived space surrounding the viewer.

The best way to think of the method is as follows:

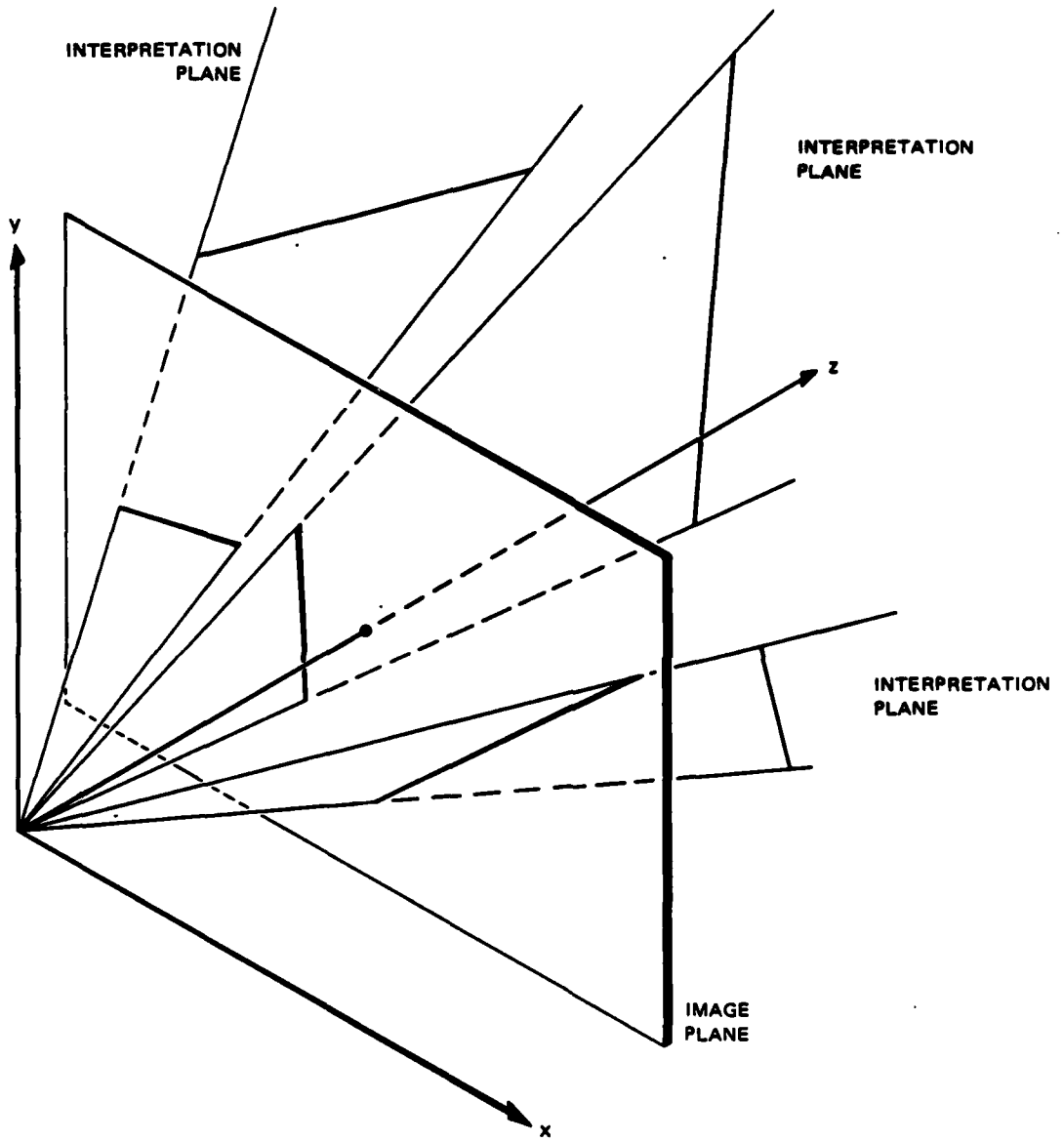
- A basis has six degrees of freedom (two degrees of freedom for each basis vector).
- Each line supplies one constraint. That is, each line constrains one of the basis vectors to a one-parameter family of vectors.
- The space of possible bases, therefore, is three-dimensional.
- The optimum basis is the one that is "most orthogonal." There will be two of these, in general.

2.1. The Most Orthogonal Basis

A system of three nonparallel, noncoplanar lines (orthogonal or not) defines a **basis** for a three-dimensional vector space. The goal is to find the basis that simultaneously is "most orthogonal" and is consistent with (i.e., explains) the two-dimensional pattern. This requires two elements: (1) a way to represent and generate the set of possible bases, and (2) a precise definition of the intuitive notion of "most orthogonal."

Let us call a pattern of three noncollinear, two-dimensional line segments (such as those shown in Figures 1 and 2) a **configuration**. We are not concerned with the length or the endpoints of the line segments. All collinear segments are considered identical. A configuration is assumed to be the result of a perspective projection of three lines in three-dimensional space (Figure 7). We will call any three such lines that produce a configuration an **admissible solution** to the configuration. Figure 7 illustrates how a configuration constrains the set of admissible solutions: the three lines of an admissible solution are constrained to lie in three planes determined by the line segments in the configuration. These planes are called **interpretation planes**. A configuration therefore can be characterized by the unit normals of three interpretation planes: (ϕ_1, ϕ_2, ϕ_3).

Clearly, the distances of the lines from the viewer are irrelevant. The lines are only required to have certain orientations and to lie in certain interpretation planes. We therefore consider all admissible solutions of a particular configuration consisting of lines of the same orientation to be equivalent. A class of equivalent admissible solutions defines an admissible basis. The basis consists of three unit vectors rooted at the origin (the center of projection), lying in the interpretation planes,



NOTE: Indicated lines in the image plane are the configuration; lines in the interpretation planes are an admissible solution.

Figure 7: Admissible Solutions

and parallel to the respective lines in the admissible solutions. These vectors can be generated in the standard, viewer-centered coordinate system. (This coordinate system is chosen such that the origin is at the projection point; the x, y, and z axes are in the directions right, up, and forward with respect to the observer; and the image plane is the plane $z = 1$. [1])

Let the three basis vectors be denoted by $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$. Remember that $\mathbf{e}_1$ lies in ϕ_1, etc. We can write a basis vector $\mathbf{e}$ in ϕ as a function of a scalar θ; for example, $\mathbf{e}_i(\theta)$ is in plane ϕ_i and at angle θ from the plane $z = 0$ (see Figure 8). The algebra for deriving this function is straightforward but somewhat tedious (see appendix).

We can now represent the set of admissible bases consistent with a configuration (ϕ_1, ϕ_2, ϕ_3):

$$S = \{[\mathbf{e}_1(\theta_1), \mathbf{e}_2(\theta_2), \mathbf{e}_3(\theta_3)] : -\pi < \theta_1, \theta_2, \theta_3 \leq \pi\}$$

Generating elements of this set is simply a matter of generating and substituting values for θ_1, θ_2, and θ_3.

The "orthogonality" of an admissible solution can be stated in a natural way as a triple product:

$$V = \mathbf{e}_1 \cdot (\mathbf{e}_2 \times \mathbf{e}_3)$$

This equation gives the volume of a parallelepiped associated with the three basis vectors (Figure 9). It is sometimes called the box product. The triple product has a maximum (or minimum) value of 1 (or -1) only when the vectors constitute an orthogonal basis. In the first case they form a left-handed basis, and in the second case a right-handed one. [2] The triple product has a value zero only when the three basis vectors are coplanar (i.e., linearly dependent).

We can find the most orthogonal basis by searching the three-dimensional space of admissible solutions for those with maximum or minimum V. In practice, there seem to be a unique minimum and maximum when an orthogonal solution is possible, and these extrema can be reached by the method of steepest ascent (or descent) from an arbitrary starting position. There is currently no proof of these conjectures, but they have held true for many different examples.

Figure 10 shows the starting point ($\theta_1 = \theta_2 = \theta_3 = 0$; i.e., the flat parallelepiped lying in the image plane), two intermediate solutions, and the final, optimal solution. The figures are produced by constructing parallelepipeds from the bases, centering them at the point $\langle 0, 0, 3 \rangle$ in the viewer coordinate system, and projecting them into the image plane with hidden-line removal. The initial parallelepiped (shown in (a)) has zero volume because all the vectors lie in one plane. The next two (shown in (b) and (c)) have successively larger volumes (hence the associated bases are more orthogonal). The final parallelepiped (shown in (d)) is actually a cube, and its associated basis is truly orthogonal. Figure 11 puts the solution in context.

[1] Note that this is a left-handed coordinate system.
[2] Because we begin with a left-handed viewer coordinate system, we apply the left-hand rule when computing the cross product.

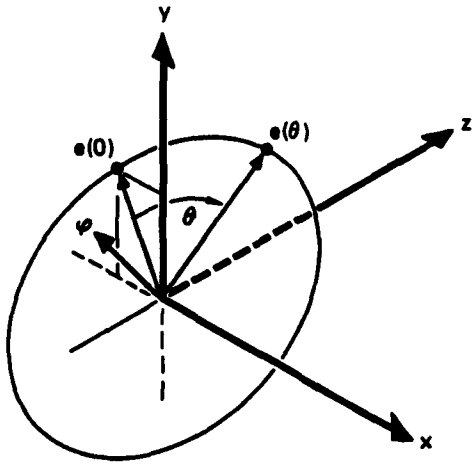


Figure 8: Basis Vector in an Interpretation Plane

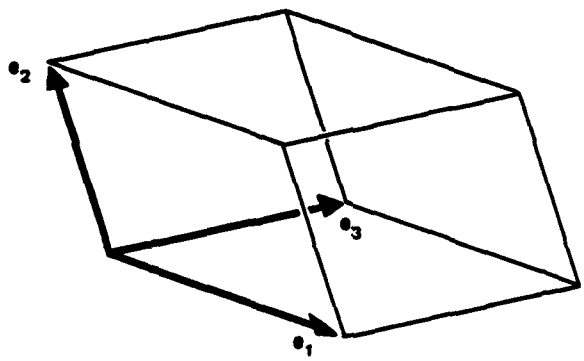


Figure 9: Parallelepiped Associated with Basis Vectors

At this point a discussion of the nature of the interpretation produced by this model is appropriate. It is a *scene-centered* (or object-centered) interpretation in the sense of orientation; that is, it decouples the natural orientation of the scene (or the object) from that of the viewer. It is a *viewer-centered* interpretation in the sense of position; that is, the origin of the most orthogonal basis is the same as the natural origin of the viewer (the center of projection).

2.2. Two Solutions: Which to Choose?

There are two ways to choose a most orthogonal basis: we can either maximize or minimize V. As mentioned above, a maximum V implies a left-handed basis, and a

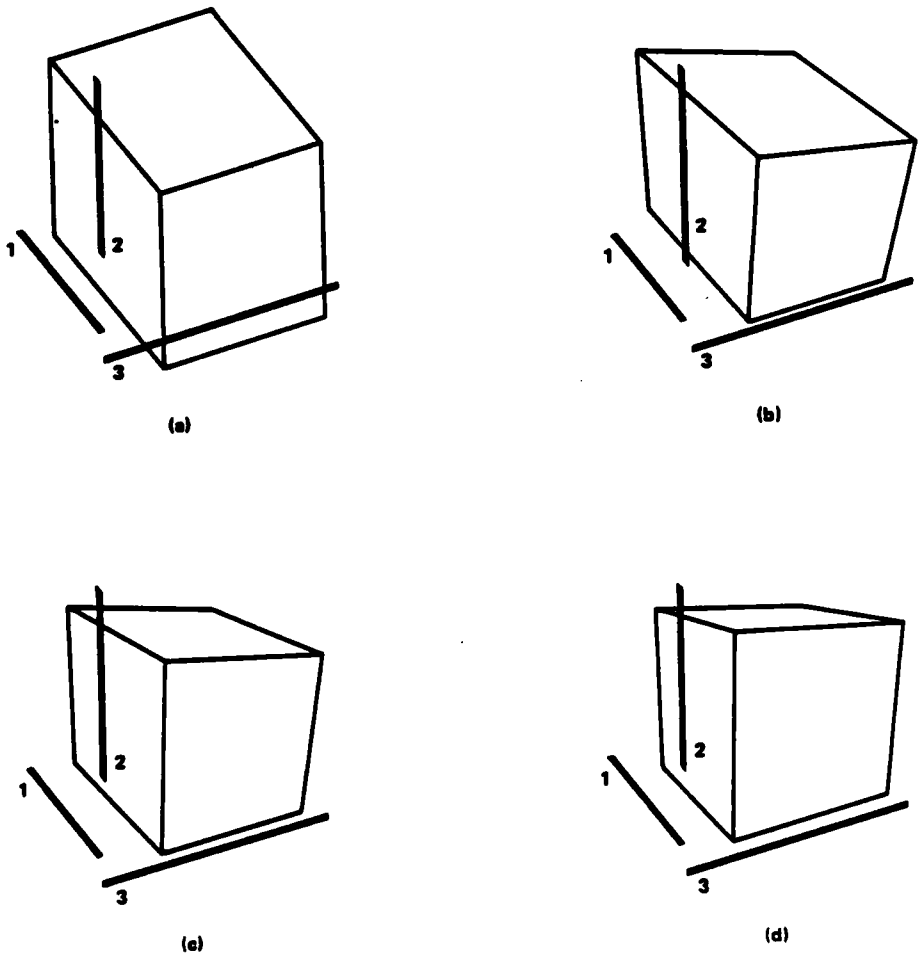


Figure 10: An Example

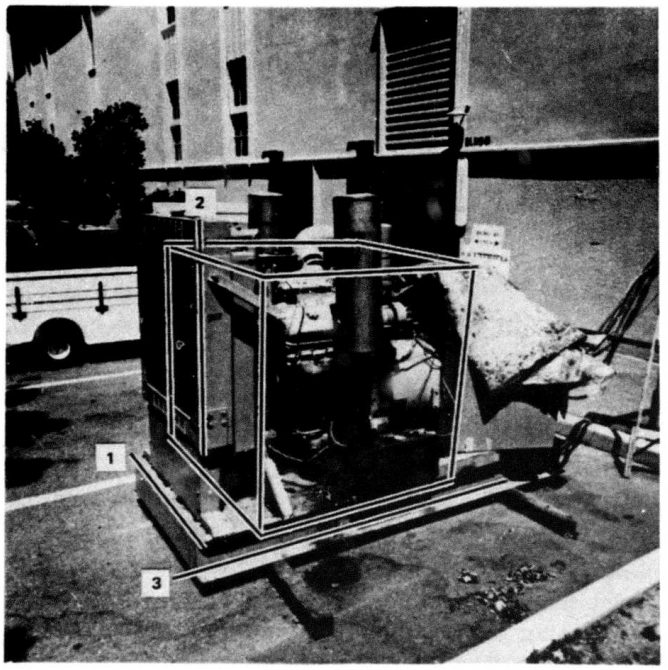


Figure 11: The Interpretation of Figure 10d in Context

minimum V a right-handed basis. The two solutions actually represent perceptual reversals. Is there any reason to choose one or the other? The handedness property is not significant because it is merely an artifact of the arbitrary direction sense of the interpretation-plane normals.

In Figure 12, a configuration and two alternate solutions are shown. For some reason, solution (b) does not seem to be as "good" as (a), even though, from a strictly mathematical point-of-view, it must be. The answer is suggested in our experiment of rotating Figure 2. The "good" solution is oriented in a way that is consistent with our notion of vertical and horizontal, but the other one is not.

In the everyday world, the effect of gravity imparts a special meaning to the "vertical" direction; similarly, while there is no unique "horizontal" direction, horizontal lines are constrained to be perpendicular to the vertical direction. All everyday scenes have a natural horizon, which may or may not be directly visible. Even if it is not directly visible, the horizon is fixed by knowledge of the vertical direction, which, presumably, is available from other visual cues and from the mechanism of the inner ear. When we view a picture, we prefer it to be aligned so that the natural horizon lies across the visual field normally (i.e., horizontally in the image).

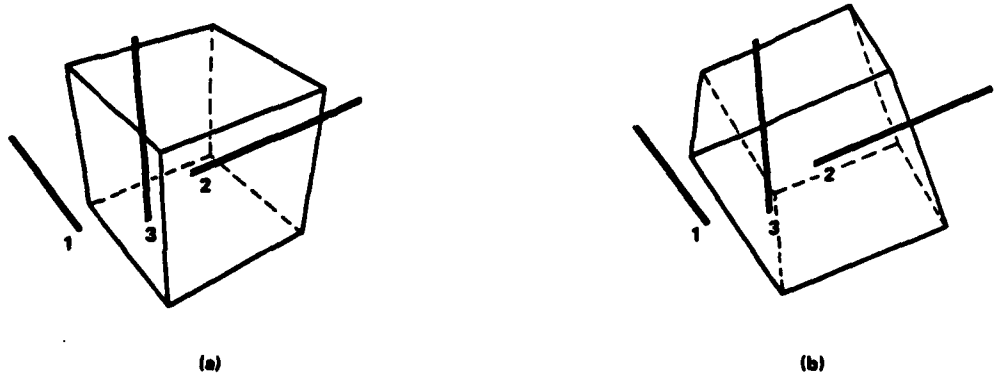


Figure 12: Two Alternate Solutions

It may be helpful to consider the relationship between the most orthogonal basis and the concept of vanishing points and lines. It is well-known that the perspective projections of parallel lines meet at common points in the image, called vanishing points. It has been shown that finding a vanishing point of a line is equivalent to finding the orientation of the line [9]. Hence, by finding the orientation of a basis vector, we determine the vanishing point of all lines parallel to it. A close examination of Figure 12 will show that, when the opposite edges of a side of the parallelepiped are extended, they intersect the extended lines of the configuration at vanishing points. Each of the two orthogonal bases therefore imply three vanishing points. Furthermore, if two of the vanishing points can be connected by a horizontal line (as in Figure 12(a)), the associated basis vectors can be interpreted as horizontal in 3-dimensional space.

2.3. Consistency

Suppose a line segment is added to a right-angle illusion. In Figure 13, three additional line segments, l_1, l_2, and l_3 are shown. Lines l_1 and l_2 seem to "fit" the rest of the illusion, while l_3 does not. That is, l_1 and l_2 can be interpreted as parallel or nearly parallel to at least one of the basis vectors, but l_3 cannot. In terms of vanishing points, we could consider all possible vanishing points of a line. If it had a possible vanishing point close to a vanishing point of a basis vector, it could be interpreted as parallel or nearly parallel to the basis vector.

Accordingly, given a basis $[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$ and a line segment l with possible interpretations $\mathbf{e}(\theta)$, we can state a estimate of l's consistency with the basis as:

$$C = \{\max_{\theta}(|\mathbf{e}(\theta) \cdot \mathbf{e}_i| : i = 1, 2, 3\} .$$

Each element of C is the absolute value of the cosine of the minimum possible angle between one of the basis vectors and a three-dimensional line that projects to l. If

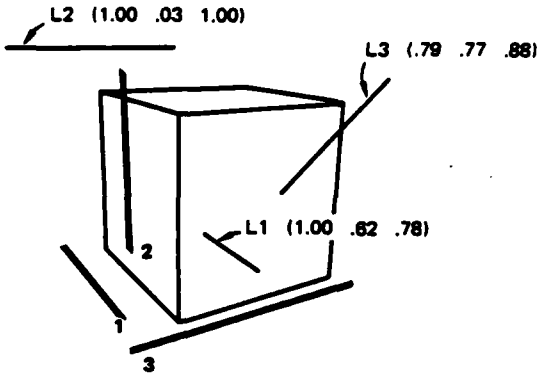


Figure 13: Consistencies of Three Lines

l can be interpreted as being parallel to a basis vector, the corresponding element of C will be one; otherwise, it will be less than one. The consistency values for l_1, l_2, and l_3 are shown in Figure 13. Note that l_1 is consistent with $\mathbf{e}_1$ only, but l_2 is consistent with both $\mathbf{e}_1$ and $\mathbf{e}_3$. Because l_2 is on the horizon, it intersects both of the horizontal vanishing points. Line l_3 is not very consistent with any basis vector.

This notion of consistency points to a way of finding the best interpretation for a large collection of lines, only some of which form a natural basis. One approach would be to select random triples of lines, solve for the most orthogonal basis, calculate the consistencies of the other lines, and choose the basis that was in some sense most consistent [10]. In the end, some lines would be inconsistent with the chosen basis and should be interpreted as having unknown orientations. A similar approach could be used to segment a scene into groups of lines that are related by virtue of being consistent with a particular basis.

It is quite possible for a configuration to lead to a most orthogonal basis that is not *actually* orthogonal (for example, three line segments separated by very acute angles, such as in Figure 4). In such a case, the method will yield a solution with $|V| < 1$. A nonorthogonal solution should probably be rejected. Of course, an orthogonal solution may also be incorrect (in the sense that it does not explain three lines in a scene correctly, because the lines are not really orthogonal). The important point is that, *given no more information than what is included in a single configuration*, the most orthogonal interpretation is reasonable.

3. Conclusions

The computational model presented in this paper is radically different from a widely prevailing view (e.g., see Marr [11]) that can be paraphrased as follows:

- Largely static, unintelligent processes convert an image into a collection of tokens, which comprise a discrete, explicit encoding of the information in the image (Marr's primal sketch).
- Evidence for local properties of surfaces (depth-from-viewer, orientation, curvature, reflectance, etc.) is extracted from the primal sketch by more-or-less autonomous processes (stereo, motion, shape-from-contour, shape-from-shading, etc.).
- This evidence is collected into a "2.5D" representation of the scene, meaning, an integrated description of surfaces in the coordinate system of the viewer.
- Instances of objects are found in the 2.5D representation (e.g., generalized cylinders).
- Finally, a description of the scene is constructed in terms of these objects.

In the model presented here, *there is no 2.5D sketch.* Furthermore, instead of a multiplicity of processes producing local, viewer-centered estimates, a single process produces a partial, scene-centered representation *directly.* The primal sketch retains its role, albeit in a more modest form — it essentially reduces to line-finding. This model is unconcerned with specific surfaces and objects. Instead, by producing a natural basis, it estimates a global property of the entire space surrounding the viewer.

This approach is most closely related to recent research on shape from contour [9], [12], [13], [14]. The general idea that relates this research is to backproject image contours onto planes of different orientations, and to choose, as the interpretation, the plane that simplifies some backprojected property. Several measures of simplicity are suggested. For example, Brady and Yuille use compactness, defined as the ratio of the area of the backprojected contour to the square of its perimeter; Barnard uses the uniformity of backprojected curvature; Witkin uses the degree of uniformity of the distribution of backprojected tangent directions. The model presented here yields interpretations not of the orientation of planes, but of space itself. Nevertheless, the philosophy is the same: to choose the most simple backprojection — in this case, simple in the sense of most orthogonal.

The Gestalt view of perception holds that percepts that are simple are preferred over those that are not. The modern version of Gestalt is that the percepts that can be most economically encoded are the ones preferred [15]. It is interesting that an orthogonal basis can be more economically encoded that a general one; that is, an orthogonal basis is **more redundant** than a general one. An orthogonal basis can be specified by any two of its basis vectors plus an indication of its handedness.

A general basis, however, requires a complete specification of all its basis vectors. The results in this paper are, therefore, consistent with, and lend support to, the information-theoretic version of Gestalt theory.

Of course, the model presented here is extremely simple and can in no way be considered a complete model of visual perception. Nevertheless, I feel that it does illustrate an important principle that is very likely to be used in human perception. Much work remains to be done to generalize and extend the model. The discussion of consistency in Section 2.3 points to one kind of generalization. The case of closed figures such as Figure 3 can be explained by an extension of the model, and this is a topic of current research. It remains to be seen whether the approach can be applied to curved contours and surfaces.

REFERENCES

1. Attneave, F. and Frost, R., The determination of perceived tridimensional orientation by minimum criteria, *Perception & Psychophysics*, vol. 6, no. 6B, December, 1969, 391-396.
2. Perkins, D., Visual discrimation between rectangular and nonrectangular parallelopipeds, *Perception & Psychophysics*, vol. 12, no. 5, 1972, 396-400.
3. Ittelson, W. H., **The Ames Demonstrations in Perception**, Hafner Publishing Company, New York, 1968.
4. Gregory, R. L., Will seeing machines have illusions?, in **Machine Intelligence 1**, N. L. Collins and D. Michie, Eds., American Elsevier, New York, 1967, 169-177.
5. Mackworth, A. K., How to see a simple world: an exegesis of some computer programs for scene analysis, in **Machine Intelligence 8**, N. L. Collins and D. Michie, Eds., American Elsevier, New York, 1977, 510-537.
6. Roberts, L.G., Machine perception of three-dimensional solids, in **Optical and Electro-Optical Information Processing**, J. T. Tippett et. al., Eds., M.I.T. Press, 1965.
7. Mackworth, A. K., Interpreting pictures of polyhedral scenes, *Artificial Intelligence* **4**, 1973, 121-137.
8. Kanade, T., A theory of origami world, *Artificial Intelligence* **13**, 1980, 279-311.
9. Barnard, S. T., Interpreting perspective images, *Artificial Intelligence* **21**, 1983, 435-462.
10. Fischler, M. A. and Bolles, R. C., Random-sample consensus: A paradigm for model-fitting with applications to image analysis and automated cartography, *Commun. ACM*, June 1981, 381-395.
11. Marr, D., **Vision**, W. H. Freeman, San Francisco, 1982.
12. Kanade, T., Recovery of the 3-D shape of an object from a single view, *Artificial Intelligence* **17**, 1981, 409-460.
13. Witkin, A. P., Recovering surface shape and orientation from texture, *Artificial Intelligence* **17**, 1981, 17-45.
14. Brady, M. and A. Yuille, An extremum principle for shape from contour, *Proceedings of IJCAI-83*, Karlsruhe, West Germany, August 8-12, 1983, 969-972.
15. Rock, I., **The Logic of Perception**, M.I.T. Press, Cambridge, Massachussetts, 1983.

A. Derivation of Functional Description of Basis Vectors

Given an interpretation plane represented by its unit normal

$$\phi = \langle \phi_x, \phi_y, \phi_z \rangle$$

we want to find an expression for the set of unit vectors in ϕ (Figure 8):

$$\{\mathbf{v} = \mathbf{v}(\theta) : -\pi < \theta \leq \pi\}.$$

We will develop and solve a system of three nonlinear equations in three unknowns: the components of $\mathbf{v}$.

The vector $\mathbf{v}(0)$ lies in the plane $z = 0$. We can impose an arbitrary directional sense to θ with

$$\mathbf{v}(0) \times \mathbf{v} = \phi \sin\theta. \tag{1}$$

This equation must hold, because $\mathbf{v}$ is perpendicular to the vector ϕ. (Refer to Figure 8. Remember that, because we use a left-handed coordinate system, we must apply the left-hand rule for a geometric interpretation of the vector cross product.)

Because $\mathbf{v}$ is a unit vector, it must satisfy

$$|\mathbf{v}| = 1. \tag{2}$$

Because $\mathbf{v}$ is in the interpretation plane ϕ, it must satisfy

$$\mathbf{v} \cdot \phi = 0. \tag{3}$$

Our system of equations is (1), (2), and (3). We will solve for $\mathbf{v}$ by first using (1) to get a simple expression for v_z, then substituting this in (2) and (3), eliminating v_y, and finally solving the resulting quadratic equation for v_x.

Let

$$D = \sqrt{\phi_x^2 + \phi_y^2}.$$

$\mathbf{v}(0)$ must be of the form:

$$\mathbf{v}(0) = (\frac{1}{D})\langle -\phi_y, \phi_x, 0 \rangle.$$

This is because it must simultaneously be in the direction

$$\phi \times \langle 0, 0, 1 \rangle = \langle -\phi_y, \phi_x, 0 \rangle$$

and satisfy

$$v_x^2 + v_y^2 + v_z^2 = 1.$$

Expanding (1), we get

$$\begin{aligned} \mathbf{v}(0) \times \mathbf{v} &= (\tfrac{1}{D})\langle -\phi_y, \phi_x, 0\rangle \times \langle v_x, v_y, v_z\rangle \\ &= (\tfrac{1}{D})\langle \phi_x v_z, \phi_y v_z, (-\phi_y v_y - \phi_x v_x)\rangle \\ &= \langle \phi_x \sin\theta, \phi_y \sin\theta, \phi_z \sin\theta\rangle. \end{aligned}$$

From the first component, we obtain our expression for v_x:

$$v_x = D \sin\theta. \tag{4}$$

Substituting (4) into (2) and (3) and expanding yields

$$v_x^2 + v_y^2 + D^2 \sin^2\theta = 1 \tag{5}$$

and

$$v_x\phi_x + v_y\phi_y + D(\sin\theta)\phi_z = 0. \tag{6}$$

Solving (6) for v_y, substituting in (5), and collecting terms yields a quadratic in v_x:

$$D^2 v_x^2 + 2\phi_x D(\sin\theta)v_x + D^2(\sin^2\theta)(\phi_y^2 + \phi_z^2) - \phi_y^2 = 0. \tag{7}$$

We solve this for v_x:

$$v_x = (\frac{1}{D})[-\phi_x\phi_z(\sin\theta) \pm \sqrt{(\sin^2\theta)(\phi_x^2\phi_z^2 - D^2(\phi_y^2 + \phi_z^2)) + \phi_y^2}]. \tag{8}$$

Now that we have expressions for v_x (8) and v_z (4), we can easily solve for v_y using (2).

Equation (7) has two solutions; the problem of which one to use can be resolved by observing that equation (3) is satisfied for two interpretation planes: ϕ and $-\phi$. This ambiguity results in the two solutions. Since the choice between ϕ and $-\phi$ is arbitrary, we can choose one and then use the appropriate form of (8).

APPENDIX B

Spatial Reasoning from Line Drawings of Polyhedra
By: Thomas M. Strat

SPATIAL REASONING FROM LINE DRAWINGS OF POLYHEDRA

Thomas M. Strat

SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025

Abstract

A method is presented for transforming a set of line drawings of a polyhedral scene into a representation that embodies the three-dimensional structure of the scene. The line drawings are first converted to machine-readable form and then back-projected to acquire a wire frame skeleton of the scene. A novel three-dimensional constraint propagation scheme is then employed to transform the wire frame to a description of the solid objects which compose the scene. This process has applications in computer-aided design as well as in machine understanding of multiple images. The paper concludes with a discussion of issues related to achieving the same result from a single view.

1. Introduction

Machines that must reason about or function in a three-dimensional world must be equipped with models of objects in that world. A multitude of representations has been devised for three-dimensional models [1], yet the specification of individual models can be a tedious undertaking. This paper examines methods for computing a three-dimensional model of a particular class of objects from a particular form of input—polyhedral objects from line drawings.

Researchers in computer-aided design have produced numerous systems that manipulate models of solid objects to assist in the design, analysis, or fabrication of everything from machine parts to factories. The act of specifying a model is one of the most difficult tasks associated with these systems.

In an interactive image-understanding system, there are several sources of line drawings. One can envision a very competent line-finder that automatically extracts the line drawings of selected objects. Alternatively, the user can specify the lines in an image by pointing at their endpoints with a mouse or other input device. A third possibility is for the user to draw the figures freehand or with mechanical assistance. Whatever the means of entry, the objective is to produce a three-dimensional sketch that captures the volumetric nature of the objects.

This paper is concerned with deriving the representation geometrically, as opposed to using model-based representations. It is divided into two parts: The first presents an algorithm for deriving a volumetric description of a polyhedral scene when multiple views are available; the second part explores the problem of accomplishing this task when only one line drawing is available.

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract No. MDA 903-83-C-0027.

2. Multiple Views

The algorithm to be described solves for a three-dimensional description of a scene when several views are available. The overall process can be thought of as accepting a set of line drawings of a scene as input and providing as output a display of the object from any angle, with all hidden lines removed. The algorithm consists of four sequential modules called input, projection, wire frame, and display.

The required input is a set of two or more line drawings and the angular relationships among them. A line drawing is restricted to be the projection (orthographic or perspective) of a polyhedron from a particular vantage point and, as a result, is a collection of straight line segments.

The input module is responsible for producing a data structure that specifies the positions of all lines and their endpoints in a line drawing. Its actual form will vary with the source of the line drawing, as different input processes dictate different procedures for constructing the data structure.

The projection module computes the three-dimensional coordinates of vertices and edges that may have given rise to the endpoints and lines in the drawings. The output of the projection module is in the form of a three-dimensional wire frame, which is represented as a list of vertices and edges. The computation is carried out by back-projecting the points in each line drawing and determining their points of intersection.

Next in the pipeline is the wire frame module, which is the most interesting of the four. Its task is to derive the solid object that corresponds to the given wire frame. It employs a Waltz-style constraint propagation scheme [8], but differs significantly by assigning labels to spatial regions and propagating them throughout the three-dimensional structure, in contrast with propagation across a two-dimensional line drawing. Only two labels are allowed (SOLID and HOLE), and a consistent labeling is usually achieved very quickly.

The display module uses the labeled output of the wire frame module to produce a display of the object, with hidden lines removed. As will be seen shortly, the hidden-line algorithm is somewhat unusual in the way it takes advantage of the label information in the wire frame.

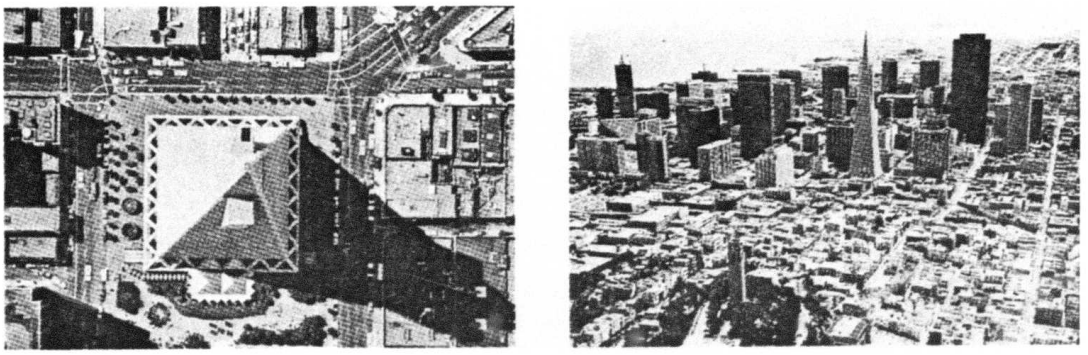


Figure 1: Photographs of the Transamerica Building

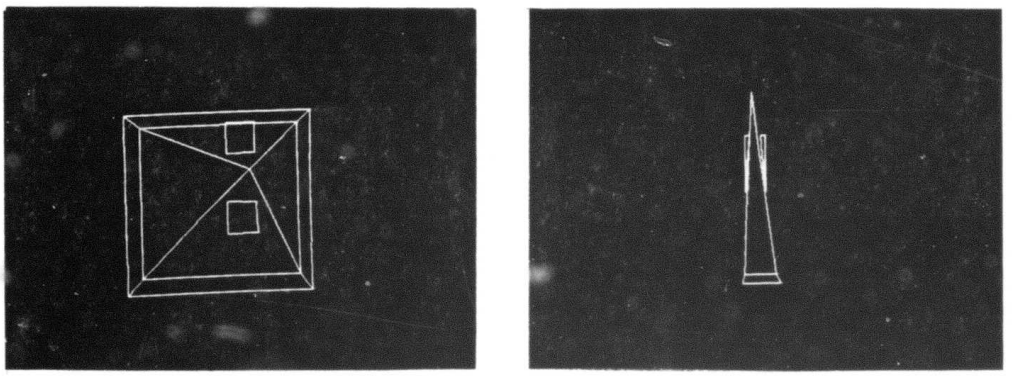


Figure 2: Line Drawings of the Transamerica Building

2.1. The Input Module

The input module is actually a set of alternative modules. The one to be used depends on the source of the line drawing. In all cases, a **line drawing** is defined to be composed of a set of line segments, to be referred to as **lines**, and their points of intersection, to be referred to as **endpoints.** Lines are restricted to intersect only at their endpoints. No curves are permitted since the line drawing is assumed to be the projection of a polyhedron. Furthermore, all edges of an object, visible or not, must appear in all line drawings. That is, hidden lines, which are normally represented as dashed lines in engineering drawings, are to be depicted like any other line, since the algorithm does not accord them any special treatment.

The most direct means for specifying the line drawing would be to provide the coordinates of the endpoints explicitly. Another procedure is envisioned that allows specification of line drawings in a more natural way.

In one scenario, the user will sketch or trace a line drawing directly into the system by means of a pointing device, such as a mouse or graphics tablet. Inaccuracies inherent in this procedure must be resolved before the data are passed to the projection module. Sugihara presents a method for identifying incorrect line drawings and correcting vertex position errors [7]. Regardless of the method used, the input module provides a data base containing the endpoints, lines, and viewpoint for each line drawing. As an example, Figure 1 shows two images of the Transamerica Building in San Francisco. The line drawings of Figure 2 were obtained by tracing the edges of the building with a mouse-controlled cursor. The camera models of each image were computed on the basis of ground truth data obtained from a map of San Francisco.

2.2. The Projection Module

Given the data base specifying a set of line drawings of a scene and the associated camera models, the projection module determines the wire frame of the scene that could have given rise to those drawings. A **wire frame** is a set of vertices and edges, where an **edge** is the intersection of two faces of an object, and a **vertex** is the intersection of two edges. The algorithm follows closely that of Wesley and Markowsky [9].

In the first phase, the vertices of the wire frame are computed. Figure 3 illustrates the geometry involved. Any vertex of the wire frame is constrained to lie on the line connecting the viewpoint and the endpoint that is the projection of the vertex in the line drawing. Such lines are constructed for every endpoint in every line drawing. The intersections of these lines are computed and entered as vertices of the wire frame. Note that this procedure may find vertices that are not in fact vertices of the

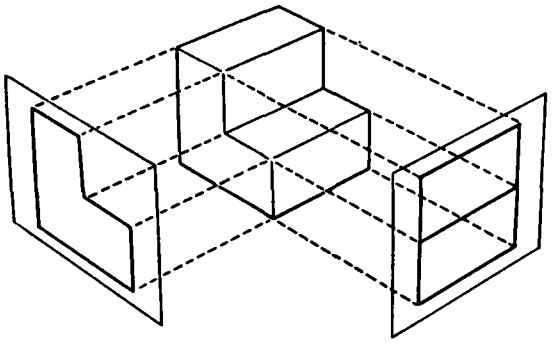


Figure 3: The geometry of backward projection.

wire frame. Some of these vertices are identified in later stages of processing and discarded; the remainder are the result of alternative legal interpretations of the line drawings. It is also possible that some real vertices may be missed, but, fortunately, they can be found during the second phase of the projection module's operation.

In Phase two, the edges of the wire frame are found. Two vertices are connected by an edge only if that edge is consistent with all views provided. An edge is consistent with a view if that edge projects to a line in the view, to a set of continuous colinear lines, or to a single endpoint. When all such edges have been found, the edges are checked for internal intersections. Any such intersections are the missing vertices of Phase one and are added to the data base. Just as extra vertices may have been found earlier, extra edges may arise for the same reasons.

In Phase three, any vertex with fewer than three incident edges is eliminated. (Realizable solid objects always have at least three edges meeting at any vertex.) Any accompanying edges are also removed and the pruning is continued until a stable configuration is reached. Usually, however, there are no vertices to be removed in this manner.

At this point, a wire frame has been computed that is guaranteed to encompass all the edges and vertices of the object. If the set of line drawings provided determines the object uniquely, the wire frame will correspond exactly to the wire frame of the object. If the line drawings are ambiguous, the wire frame may contain vertices and edges present in one interpretation but absent in others. The ambiguous case can be accommodated by invoking the wire frame module for each of the possible interpretations. Those found to be inconsistent can be disregarded; those found to have a legal interpretation can be construed as alternative solutions. The remainder of this paper assumes that the wire frame has been determined uniquely.

2.3. The Wire Frame Module

The input to the wire frame module is a data structure representing a wire frame that contains only edges and vertices that correspond to true edges and vertices of the underlying scene. The module's job is to find out which regions are occupied by

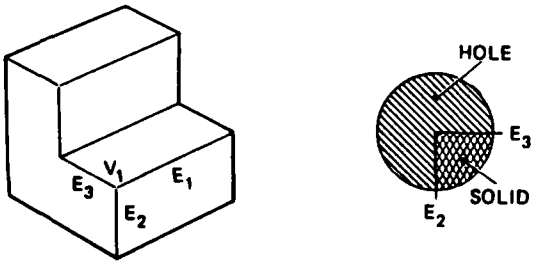


Figure 4: Spoke diagram of edge E_1.

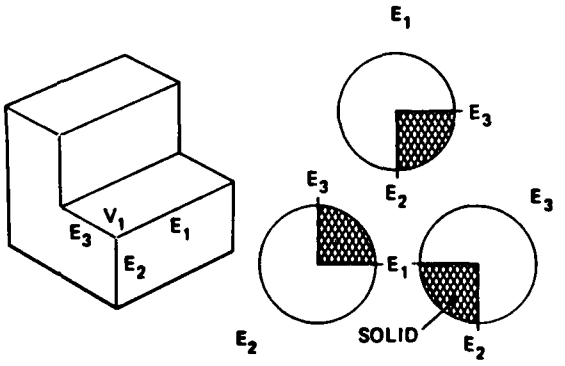


Figure 5: Propagation of Intravertex Constraints.

solid matter and which are not, relative to the wire frame. Its basic tool for performing this reasoning is the **spoke diagram** (Figure 4). The spoke diagram is an edge-on view of a vertex. The spokes are the projections of the edges at a vertex onto the plane that is perpendicular to the selected edge. The spoke diagram in the figure is the view along edge E_1 toward vertex V_1, such that E_1 itself projects out of the drawing. The sector between two spokes represents the solid angle defined by the two edges corresponding to the two spokes and the selected edge. The solid angle must either be filled completely with matter or be completely void of matter, because boundaries between matter and space can occur only at faces and all faces are bounded by edges. Therefore, each sector can be labeled by either SOLID or HOLE to reflect this choice. The task of the wire frame module is then to assign a label of SOLID or HOLE to every such solid angle, as defined by the wire frame.

As mentioned earlier, the wire frame module is a constraint propagation algorithm. Three separate processes serve to constrain and propagate the labelings:

1. **Intravertex constraints**—These serve to propagate labels (SOLID or HOLE) among the spoke diagrams at a given vertex, as in the example of Figure 5. Here an assignment of SOLID to the sector between the spokes corresponding to edges E_2 and E_3, in the spoke diagram of E_1 at

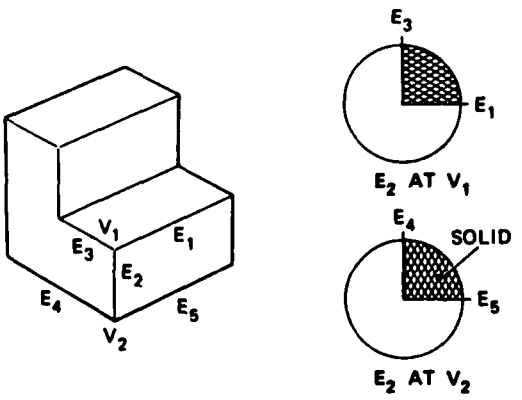


Figure 6: Propagation of Intervertex Constraints.

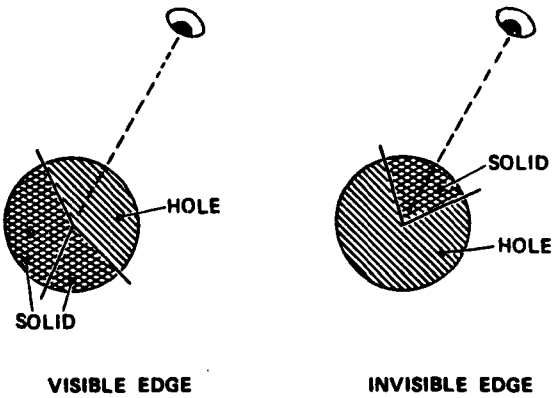


Figure 7: Hidden-line removal.

vertex V_1, can be propagated to constrain the other spoke diagrams at V_1. In the spoke diagram of edge E_2, the sector between edges E_1 and E_3 must also be labeled SOLID. The same applies to edge E_3. Corresponding sectors must be labeled identically because they are representations of the same volume in space. Care must be taken to account for sectors wider than 180°.

2. **Intervertex constraints**—These serve to propagate labels from one vertex to another, as in Figure 6. Here the newly assigned label of SOLID to the sector between E_1 and E_3, in the spoke diagram of edge E_2 at vertex V_1, can be propagated along E_2 to constrain the spoke diagrams at V_2. In this case the sector between E_4 and E_5 must also be SOLID. The two pairs of edges define a dihedral angle along E_2. This type of constraint propagation is always valid because the state can change between SOLID and HOLE only at a face. If a face occurred somewhere along E_2, it would intersect with E_2 at a point other than one of its endpoints, which is precluded by the definition of a wire frame.

3. **Vertex Labeling Constraints**—Some potential labelings of a spoke diagram are not legal. It is conceivable to determine the set of legal labelings for trihedral vertices, tetrahedral vertices, etc., but in practice this becomes unwieldy. As implemented, the wire frame module applies a somewhat weaker constraint. It prohibits any labeling that is all SOLIDs or all HOLEs. Since such an assignment would render the edge nonexistent, it could not be a legal interpretation of the wire frame. In practice, this rather weak constraint has generally proved satisfactory.

Unfortunately, there are numerous special cases that arise during execution. The intravertex constraints must distinguish between angles greater or less than 180° to ensure proper labeling. The intervertex constraints must be applied properly when the spoke diagrams at each endpoint do not coincide. Extra spokes and missing spokes are two such cases.

The wire frame module is a control structure for propagating these three constraints throughout a wire frame. For efficiency, the implementation actually applies the three constraints simultaneously. It includes checks for completion and inconsistencies. Although a given wire frame may be ambiguous, (*i.e.*, allow more than one interpretation), the algorithm is guaranteed to terminate. Each step assigns a label, does nothing, or detects an inconsistency and quits. Once a label is assigned, it is never removed. The algorithm proceeds until all sectors have labels or no change has been detected through a complete iteration. If some sectors remain unlabeled, the algorithm is continued separately for each possible labeling (SOLID or HOLE) until a completely labeled wire frame is obtained.

2.4. The Display Module

The display module uses the labels computed by the wire frame module to eliminate lines hidden from view. Most of the lines to be eliminated can be readily identified by the process illustrated in Figure 7. For each edge, the direction to the viewpoint is computed and that ray is superimposed on the spoke diagram. If the ray pierces a SOLID sector, that edge is a rearward-facing edge and is not displayed. If the ray falls into a HOLE sector, further processing is needed. The projection of the edge is checked for intersection with the projection of all other visible edges. If no such intersection exists, the edge is displayed. If an intersection is found, the intersecting edge is examined to determine which side of the edge is occluded (or if both are). The edge is split at its point of intersection and each part is handled in the same manner recursively. It should be noted that this algorithm will fail to eliminate certain occluded edges if their projections do not intersect with any other projected edges. A less efficient search would be required to eliminate these.

Returning to the example of the Transamerica Building of Figure 1, a wire frame was obtained from the line drawings and was subsequently processed by the wire frame module. Figure 8 shows a perspective view of the result, with hidden lines removed, indicating realization of the correct wire frame and the successful assignment of solid material relative to it.

2.5. Summary

The class of objects that may be modeled is not as restrictive as it may seem at first glance. Any polyhedron is permissible and may contain arbitrary concavities and holes. Curves may be approximated by a number of line segments. Several polyhedra may be juxtaposed in any manner.

The line drawings may be either orthographic or perspective, and from any vantage point. Accidental alignments pose no par-

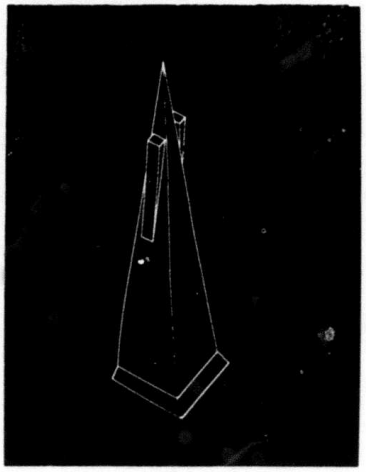


Figure 8: Hidden-line removal on the Transamerica Building.

ticular problem. A unique wire frame will be constructed if any drawing is in general position (*i.e.*, no vertices or edges coincide in the projection). Reconstruction of a unique wire frame is also guaranteed if corresponding lines in each view are designated.

Just as a set of line drawings may define more than one wire frame, some wire frames may define more than one object. The wire frame module will derive all possible interpretations when confronted with an ambiguous wire frame.

Stronger vertex-labeling constraints may be necessary to assure the correct interpretation of some objects. Further testing is required to refine the constraints and demonstrate the ability to derive volumetric descriptions of a wide range of polyhedral scenes.

3. Single View

We have described a procedure that makes it possible to infer the shape of an object from several views. A means for accomplishing this task from a single view is also desirable. After all, humans appear to have little trouble constructing a three-dimensional representation of an arbitrary object from a single line drawing. While this section does not offer a method for approximating human performance, it does point to some promising approaches toward that objective.

The most relevant early work on polyhedral-scene interpretation is by Mackworth [6]. His program, POLY, has provided a framework for subsequent shape-from-line-drawing methods. POLY first achieves a qualitative interpretation of a line drawing by parsing the line segments to ascertain which of them are convex, concave, or occluding. The convex and concave edges yield constraints that can often be used to determine the orientation of the polyhedral faces quantitatively. By using the now familiar technique, the orientations of faces that meet at an edge are restricted to lie on a line in gradient space that is perpendicular to the image line of that edge. Combining these constraints through triangulation often yields the orientations of all the faces.

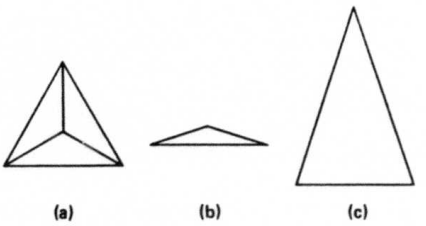


Figure 9: An ambiguous drawing.

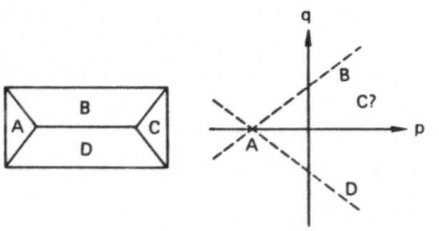


Figure 10: Another ambiguous drawing.

Unfortunately, POLY has many shortcomings that limit its competence, as pointed out by Draper [3]. He presents methods for overcoming some of POLY's limitations, yet human-level performance is still beyond reach.

One observation that dampens hope of ever producing an algorithmic solution to the problem is that human perception of line drawings can be extremely subjective. Figure 9(a) shows a triangular pyramid that can be variously interpreted as either very flat (b), or very pointed and elongated (c). In gradient space, this phenomenon manifests itself in the fact that the scale of the gradient space cannot be resolved from a line drawing algebraically without higher-level assumptions. In fact, neither the scale nor the origin of gradient space can be determined by existing methods that utilize the gradient-space representation.

Recent work by Barnard [2] addresses this issue of subjective interpretation. For instance, if one is willing to assume that the pyramid in Figure 9 is actually composed of mutually perpendicular faces, that information can be used to specify the orientations of all faces of the object uniquely. Moreover, what this accomplishes is to establish both the origin and scale of gradient space. Barnard's procedure only requires identification of three lines in the scene known to be orthogonal (they need not intersect) in order to compute these quantities. This represents a powerful tool when coupled with the purely objective approach to the interpretation of line drawings. The requirement for finding a set of mutually orthogonal lines is no great burden when many scenes of man-made objects are being examined, but one would prefer a purely general method for the subjective interpretation of line drawings. Heuristics exploiting parallelism, symmetry,

and compactness of description may provide a useful inroad.

The determination of the origin and scale of gradient space is in itself not sufficient for the interpretation of all faces in a polyhedral scene. Figure 10 shows an object and its gradient-space interpretation. Even if the orientation of face *A* is known exactly, the locations in gradient space of faces *B*, *C*, and *D* are still underdetermined. The figure is analytically ambiguous but subjectively resolvable; additional heuristics may be necessary to find the solution.

Another issue inherent in the analysis of line drawings is how best to cope with imprecise input. Line drawings may be extracted from real images or may be hand-drawn. One would prefer an algorithm that does not degenerate completely when confronted with inaccurate drawings. A drawing of an "impossible" object, that is, a drawing that does not correspond to any geometrically possible object, should be interpreted as the "closest" object that is geometrically permissible. Kanade's algorithm [4], which works through iterative minimization of errors, provides a framework for achieving this goal. One can conceive of designing a system that theoretically supports only orthographic line drawings, and using it to interpret perspective drawings. If the focal length is sufficiently large, the perspective distortion might be treated as drawing error and an approximate interpretation obtained. While the validity of this approach depends on the application in mind, it does circumvent the difficulties of a truly perspective model.

The gradient-space representation is unsuitable for analyzing perspective drawings [5]. The primary reason is the inability to capture the concept of sidedness of a plane in gradient space. Sidedness reasoning is essential to the interpretation of perspective drawings because either side of a plane may be visible, depending on the plane's location in a perspective drawing. Formalisms based on the Gaussian sphere overcome this problem. The mathematics becomes a little more complex (quadratic versus linear equations), but the two solutions to each quadratic equation, corresponding to the two sides of a plane in three-dimensional space, enable quantitative analysis of perspective scenes.

4. Summary

Recovering the shape of an object from a single line drawing of that object is a difficult problem. Further investigation is necessary to achieve human-level competence.

The algorithm presented for interpreting scenes from multiple views embodies a novel approach to a long-standing problem. The type of spatial reasoning used promises to be applicable in other situations as well. The technique may be successful when only a portion of an object is visible and may perform adequately even with inaccurate line drawings (such as those missing a line here or there). It is a local reasoning process that may be especially appropriate for supporting higher-level reasoning about solid objects.

REFERENCES

[1] Baer, A., Eastman, C., and Henrion, M., "Geometric Modeling: a Survey", *Computer Aided Design*, Vol. 11, September 1979, pp. 253-272.

[2] Barnard, S. T., "Interpreting Perspective Images", Technical Note 271, Artificial Intelligence Center, SRI International, Menlo Park, California, November 1982.

[3] Draper, S. W., "The Use of Gradient and Dual Space in Line-Drawing Interpretation" *Artificial Intelligence* 17, August 1981, pp. 461-508.

[4] Kanade, T., "Recovery of the 3-D Shape of an Object from a Single View", *Artificial Intelligence* 17, August 1981, pp. 409-460.

[5] Kender, J. R., "Shape from Texture", CMU-CS-81-102, Carnegie-Mellon University, November 1980.

[6] Mackworth, A. K., "Interpreting Pictures of Polyhedral Scenes", *Artificial Intelligence*, 4, 1973, pp. 121-137.

[7] Sugihara, K., "Mathematical Structures of Line Drawings of Polyhedrons— Toward Man-Machine Communication by Means of Line Drawings", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 5, September 1982, pp. 458-469.

[8] Waltz, D., "Understanding Line Drawings of Scenes with Shadows", *The Psychology of Computer Vision*, P.H. Winston, Ed., McGraw-Hill Book Co., Inc., New York, 1975, pp. 19-91.

[9] Wesley, M.A. and Markowsky, G., "Fleshing Out Projections", *IBM J. Res. Develop.*, Vol. 25, No. 6, November 1981, pp. 934-954.

APPENDIX C

Recovering the Camera Parameters from a Transformation Matrix

By: Thomas M. Strat

Recovering the Camera Parameters from a Transformation Matrix

Thomas M. Strat
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Abstract

The transformation of the three-dimensional coordinates of a point to the two-dimensional coordinates of its image can be expressed compactly as a 4 x 4 homogeneous coordinate transformation matrix in accordance with a particular imaging geometry. The matrix can either be derived analytically from knowledge about the camera and the geometry of image formation, or it can be computed empirically from the coordinates of a small number of three-dimensional points and their corresponding image points. Despite the utility of the matrix in image understanding, motion tracking, and autonomous navigation, very little is understood about the inverse problem of recovering the projection parameters from its coefficients. Previous attempts have produced solutions that require iteration or the solution of a set of simultaneous nonlinear equations. This paper shows how the location and orientation of the camera, as well as the other parameters of the image-formation process can easily be computed from the homogeneous coordinate transformation matrix. The problem is formulated as a simple exercise in constructive geometry and the solution is both noniterative and intuitively understandable.

1 Introduction

Homogeneous coordinates and the *homogeneous coordinate transformation matrix* are a convenient means for representing arbitrary transformations, including perspective projection in a single formalism. One such use for this matrix is as a *camera transformation matrix* that maps points in an object-centered coordinate system into the corresponding points in image coordinates according to a particular imaging geometry [7]. The camera transformation matrix has seen wide use in several disciplines. Rogers and Adams present numerous applications in computer graphics [8]. Other fields that have made use of the camera transformation matrix include stereo reconstruction, robot vision, photogrammetry, unmanned-vehicle guidance, and image understanding [5], [6], [12], [11]. Several techniques for computing the matrix have been derived, yet very little is understood about how to recover the projection parameters from the coefficients of the matrix.

When the location and orientation of the camera are known, the camera transformation matrix that models the image formation process can easily be derived analytically [1]. This model forms the basis for subsequent processing of images produced by that camera. On the other hand, when the location and orientation of the camera are unknown, the parameters of image formation must be derived from the correspondence between a set of image features and a set of object features. Images obtained from an unknown source or from cameras mounted on moving platforms exemplify contexts in which the imaging geometry may not be known.

One approach to computing the parameters of the image formation process directly is embodied in RANSAC, developed by Fischler and Bolles [3]. RANSAC computes the camera location directly from a set of landmarks with known three-dimensional locations when, in addition, the focal length and piercing point are known.

Alternatively, several methods exist for estimating the coefficients of the camera transformation matrix from the correspondence between image and object coordinates. Sutherland [10] describes a method to determine the matrix experimentally from the image by using a least-squares technique to obtain the coefficients from available ground truth data. A consideration of the experimental errors involved and a means for improving accuracy are described by Sobel [9].

The issue addressed in this paper is how to determine the imaging geometry from a camera transformation matrix that has been derived experimentally. For example, given a photograph taken by an unknown camera from an unknown location and which, moreover, may have been cropped and/or enlarged, how can we recover the camera's position and orientation and determine the extent to which the picture was cropped or enlarged? If some ground truth data are available, an established least-squares technique such as Sutherland's can be used to derive the camera transformation matrix, whereupon the problem reduces to that of computing the values of the desired parameters from the matrix. Ganapathy [4] recently published the first noniterative method for solving this problem by posing it as a set of eleven simultaneous nonlinear equations that can be solved to obtain the eleven independent coefficients of the camera transformation matrix. While the method is successful at solving for camera location and orientation, it is an algebraic one that provides little insight into the underlying geometry. The method to be described here is a geometric one that solves for the same parameters, but is posed as a simple problem in constructive geometry and allows an intuitively clear derivation.

This work has immediate application in several areas:

- Many algorithms in image understanding require knowledge of the camera parameters. These can be computed from an arbitrary photograph by using the method presented here when ground truth data is available.

- Autonomous navigation can be posed as a problem in deriving camera parameters. A cruise missile, for instance, could obtain the camera transformation matrix from a terrain model stored on board and then compute the camera parameters that define the vehicle's location and heading.

- A stationary camera viewing a robot arm workspace could determine the position and orientation of the arm. Conspicuous marking of several points on a part of the manipulator would allow their easy extraction from an image and provide the ground truth necessary for Sutherland's algorithm to ascertain the camera transformation matrix. The camera parameters can be derived from this matrix, and the location and orientation of the manipulator can then be obtained relative to the stationary camera.

2 The Camera Transformation Matrix

As indicated earlier, the camera transformation matrix can be used to model in a single formalism the effects of rotation, translation, perspective, scaling, and cropping—*i.e.*, all the variables associated with the normal imaging process. Here we review the fundamentals of homogeneous coordinate transformations that are essential for understanding the decomposition to be described. The presence of an ideal lens and the absence of any atmospheric distortions are assumed.

The imaging situation can be modeled as shown in Figure 1. The XYZ coordinate system represents the world or object-centered coordinates. The *center of projection* (the location of the lens) is shown as a point L in space. The *image plane* is a plane between the lens and the object onto which the object is projected to obtain the image. Each *image point* is that point in the image plane where the plane intersects the line connecting L with the corresponding *object point*. The UVW coordinate system is situated such that (u, v) are the image coordinates of an image point and $w = 0$ defines the image plane. The perpendicular distance between L and the image plane is the camera *focal length*, f.

In a homogeneous coordinate system, a three-dimensional point (x, y, z) is represented as a four-component row vector, (tx, ty, tz, t); the three-dimensional coordinates are obtained by dividing through by the fourth component. A point in the world is represented as a four-component row vector and its projection in the image is obtained by postmultiplying by the 4 x 4 camera transformation matrix:

$$\mathbf{x}M = \mathbf{u}$$

$$(x, y, z, 1)M = (su, sv, sw, s)$$

This homogeneous coordinate system is most useful for modeling the effects of perspective projection—further details can be found in Ballard and Brown [1].

The matrix M can be viewed as being composed of several simple transformations. While it is possible to decompose the matrix in a variety of ways, the particular decomposition chosen must capture all the degrees of freedom of the imaging geometry. The somewhat arbitrary choice used throughout this paper is shown below:

$$M = (translate)(rotate)(project)(scale)(crop)$$

$$M = TRPSC \qquad (1)$$

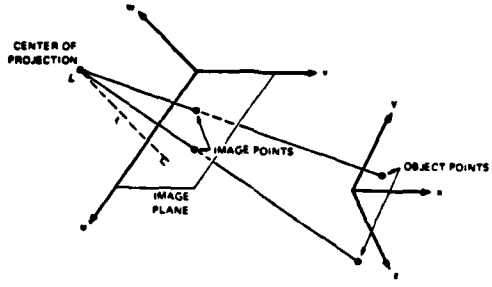


FIGURE 1 THE IMAGING GEOMETRY

Each of the component transformations can be expressed as a 4 x 4 matrix; multiplying them together produces the camera transformation matrix M. Details of the decomposition are given below.

2.1 Translation

Translation moves the image plane away from the object-centered origin. To translate the plane by (x_0, y_0, z_0) multiply by the matrix

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}.$$

2.2 Rotation

The orientation of the camera is specified by the rotation matrix R, which can be further decomposed to $R = R_x R_y R_z$, corresponding to rotation about each of the principal axes. Clockwise rotation by θ about the X axis while looking toward the origin is accomplished by

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Similarly, clockwise rotation by ϕ about the newly rotated Y axis is represented by

$$R_y = \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

and rotation by ψ about the new Z axis is given by

$$R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The first two rotations, R_x and R_y, serve to align the Z axis with the line of sight defined by the W axis. The final rotation, R_z, is within the image plane about the line of sight. Together, T and $R = R_x R_y R_z$ account for the location and orientation of the camera.

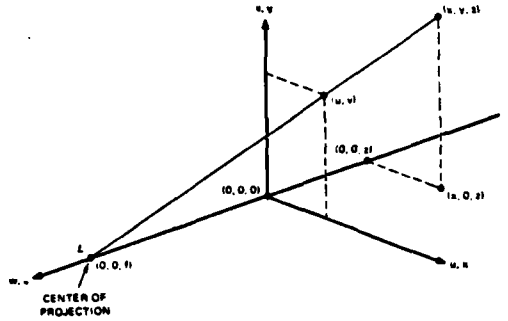


FIGURE 2 PERSPECTIVE PROJECTION AFTER ROTATION AND TRANSLATION

$z = 0$ is the image plane.

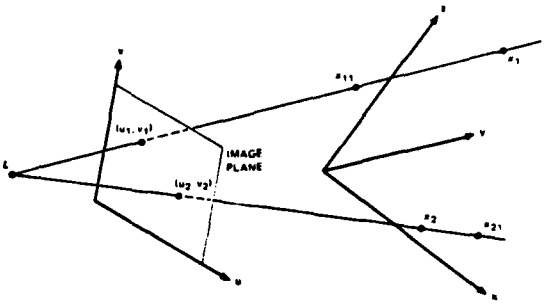


FIGURE 3 DETERMINING THE LOCATION OF THE CAMERA

2.3 Projection

P provides the distortion associated with perspective projection. Figure 2 shows the simplified imaging geometry after translation and rotation have been accounted for. The camera location L is on the positive Z axis at a distance f from the origin. The image plane passes through the origin and the world to be imaged lies behind the image plane. Analysis of similar triangles shows that the image coordinates

$$(u, v) = \left(\frac{fx}{f - z}, \frac{fy}{f - z}\right).$$

Using homogeneous coordinates, this perspective projection can be obtained by multiplying the homogeneous coordinates of the world point by the matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/f \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The resulting row vector is divided through by the fourth component to renormalize the homogeneous coordinates, and then projected orthographically onto the image plane $w = 0$ to yield the proper perspective projection of the world point.

2.4 Scaling

The image coordinates can be scaled to reflect an enlargement or shrinking of the image. Scaling by k_u and k_v in the U and V directions is achieved with

$$S = \begin{bmatrix} k_u & 0 & 0 & 0 \\ 0 & k_v & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Scaling the W axis is meaningless because the perspective projection always requires that $w = 0$.

2.5 Cropping

The effect of cropping a photograph is obtained by translating the UV coordinates within the image plane. The following matrix is used to shift the origin by (u_0, v_0):

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -u_0 & -v_0 & 0 & 1 \end{bmatrix}.$$

Note that neither scaling nor cropping affects the w and s coordinates of the homogeneous image point, so that orthographic projection and renormalization can take place after the entire transformation has been computed.

3 Recovering the Camera Parameters

The camera transformation matrix M allows representation of all eleven degrees of freedom associated with the image formation process. These camera parameters are embedded in the matrix in a way that makes their determination difficult. This section presents a simple method that recovers the various parameters associated with image formation. Its main advantages are that it is both noniterative and geometric, enabling a clear understanding of the equations involved.

The matrix M can be viewed as a function that maps world coordinates into image coordinates according to the constraints of Figure 1. For notational simplicity, we shall assume that all matrix multiplications automatically normalize the homogeneous coordinates of the resulting row vector. For example, $\mathbf{u} = \mathbf{x}M = (su, sv, sw, s) = (u, v, w, 1)$. The image formation process can then be written as

$$(u, v, 0, 1) = orthoproject(\mathbf{x}M), \tag{2}$$

where x is the homogeneous coordinate of a world point, and $orthoproject(\cdot)$ is a function that performs an orthographic projection along the w axis such that

$$orthoproject(u, v, w, 1) = (u, v, 0, 1).$$

3.1 Location of the Camera

Figure 3 illustrates the technique for finding the center of projection. First, compute M^{-1} for later use. Note that M will always be invertible because all its components in Equation 1 are clearly invertible. The location of the center of projection, L, can be determined as follows:

Choose an arbitrary world point $\mathbf{x}_1 = (x_1, y_1, z_1, 1)$ and compute $\mathbf{u}_1 = \mathbf{x}_1 M$. If we were to multiply $\mathbf{u}_1 = (u_1, v_1, w_1, 1)$ by M^{-1}, we would obtain the original $\mathbf{x}_1$. Instead, first project $\mathbf{u}_1$

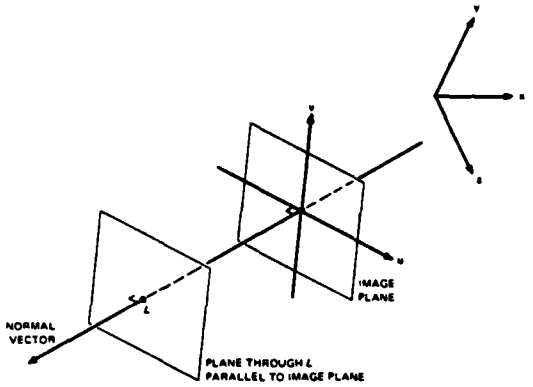


FIGURE 4 DETERMINING THE ORIENTATION OF THE CAMERA

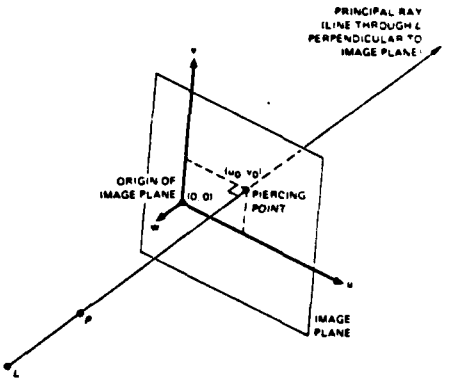


FIGURE 5 DETERMINING THE PIERCING POINT

to obtain $orthoproject(\mathbf{u}_1) = (u_1, v_1, 0, 1)$, where (u_1, v_1) are the image coordinates of $\mathbf{x}_1$. Next, backproject this image point by multiplying by M^{-1} to obtain $\mathbf{x}_{11}$. This specifies another world point, $\mathbf{x}_{11}$, which is different from the original $\mathbf{x}_1$ but constrained to lie along the line connecting $\mathbf{x}_1$ and the center of projection L. To confirm this, note that all points lying along the line connecting $\mathbf{x}_1$ and L are transformed by M to points identified by $(u_1, v_1, w, 1)$, where w varies with each point. The converse must also be true. That is, for any w, $(u_1, v_1, w, 1)M^{-1}$ specifies a point somewhere along the line connecting $\mathbf{x}_1$ and L.

Repeat the above process with another point $\mathbf{x}_2$ to obtain the point $\mathbf{x}_{21}$, which must lie on the line connecting $\mathbf{x}_2$ and L. Now $\mathbf{x}_1$ and $\mathbf{x}_{11}$, and $\mathbf{x}_2$ and $\mathbf{x}_{21}$ define two lines that pass through L; their intersection can be computed to obtain the world coordinates of L. This method will fail, of course, if either $\mathbf{x}_1$ or $\mathbf{x}_2$ lie in the image plane *or if* $\mathbf{x}_1$, $\mathbf{x}_2$ and L are colinear. Because their choice is arbitrary, valid points can always be found that allow the unique determination of L.

3.2 Orientation of the Camera

The orientation of the camera is defined by the orientation of the image plane (Figure 4). The latter can easily be established by observing that world points lying in the plane that is parallel to the image plane and that passes through the center of the lens will map to infinity in image coordinates. The only way this can happen for a finite world point is if the fourth component of the homogeneous image coordinate is zero.

Thus, if

$$(x, y, z, 1)M = (u, v, w, 0),$$

it follows that

$$M_{14}x + M_{24}y + M_{34}z + M_{44} = 0,$$

which is the equation of the plane through L parallel to the image plane. From this equation it is clear that the vector $\mathbf{n} = (M_{14}, M_{24}, M_{34})$ is normal to the image plane and parallel to the camera's direction of view.

The orientation in terms of rotations about the axes can be calculated by using spherical coordinates such that

$$\theta = \arctan \frac{M_{24}}{-M_{34}}$$

and

$$\phi = \arcsin \frac{-M_{14}}{\sqrt{M_{14}^2 + M_{24}^2 + M_{34}^2}}$$

where θ is the clockwise rotation about the X axis and ϕ is the clockwise rotation about the rotated Y' axis. The final rotation parameter, ψ, is the rotation within the image plane about the W' axis. The magnitude of ψ cannot be obtained from the normal to the image plane; instead, it requires a more complex derivation that involves determination of the piercing point and the relative scale factors. These values are derived in the following sections and the value of ψ is finally computed in Section 3.5.

3.3 Piercing Point, Principal Ray, and Cropping

Much work in image understanding requires knowledge of the *piercing point* (or *stare point*) in an image. This is the point in an image that corresponds to the world point at which the camera was aimed. *It is the* point at which the *principal ray* (the ray along which the camera was aimed) pierces the image plane (Figure 5). The principal ray is assumed to be perpendicular to the image plane.[1]

To find the piercing point, first find a point $\mathbf{p}$ along the principal ray (other than L):

$$\mathbf{p} = L + k\vec{n},$$

where k is any scalar except 0. The piercing point $\mathbf{u}_0$ is given by

$$\mathbf{u}_0 = orthoproject(\mathbf{p}M) = (u_0, v_0, 0, 1)$$

because any point along the principal ray must project to the piercing point in the image. The extent to which the image has been cropped is given by (u_0, v_0).

3.4 Focal Length and Scale

When the center of projection is held a constant distance from the scene, there is no way to tell the difference between scaling the image and varying the focal length. For example, doubling the focal length is equivalent to enlarging the picture by a factor of two. The best one can hope for is a relation between the two

[1] The image plane in some cameras used in photogrammetry is not perpendicular to the line of sight; this case, however, is not considered here.

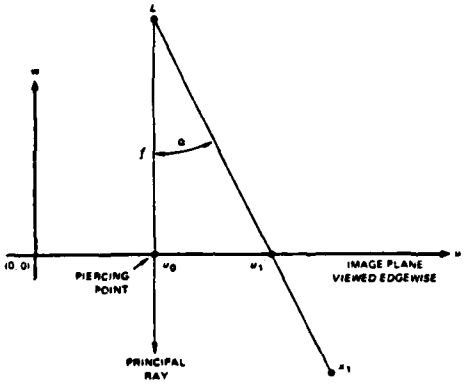


FIGURE 6 DETERMINING THE SCALE FACTOR

parameters. On the other hand, if the focal length of the camera is known, the exact scale factors can be determined.

Figure 6 shows the geometry for computing the U-component of the scale factor. First choose an arbitrary world point $\mathbf{x}_1$ (not on the principal ray) and compute its image point $\mathbf{u}_1 = \mathit{orthoproject}(\mathbf{x}_1 M) = (u_1, v_1, 0, 1)$. Conversion of these image units to world units requires dividing by the scale factor such that

$$u_1' = \frac{u_1}{k_u} \quad \text{and} \quad v_1' = \frac{v_1}{k_v},$$

where u_1' and v_1' are the distances of an image point from the image origin, measured in world units. Next compute α, the angle between the principal ray and the ray from L to $\mathbf{x}_1$, projected in the plane $v=0$. Then it is clear from the diagram that the following relation must hold

$$\tan\alpha = \frac{u_1' - u_0'}{f} = \frac{\frac{u_1}{k_u} - \frac{u_0}{k_u}}{f},$$

where k_u is the magnification of the image in the U direction. If the focal length is known, the scale factor

$$k_u = \frac{u_1 - u_0}{f \tan\alpha}$$

or if the scale factor is known,

$$f = \frac{u_1 - u_0}{k_u \tan\alpha}.$$

The computation of k_v, the V-component of the scale factor, is identical. Neither k_u nor k_v can be determined individually without knowledge of the focal length, but their ratio can be calculated from quantities derived from the matrix:

$$\frac{k_u}{k_v} = \frac{u_1 - u_0}{f \tan\alpha_u} \Big/ \frac{v_1 - v_0}{f \tan\alpha_v} = \frac{(u_1 - u_0)\tan\alpha_v}{(v_1 - v_0)\tan\alpha_u}$$

3.5 Rotation within the Image Plane

We now return to the derivation of ψ, the rotation of the camera about the W axis. This rotation is equivalent to cropping the image at an angle to the UV coordinate system. The value of ψ is found by choosing a world point and comparing its transformation under two different situations, as illustrated in Figure 7.

First, use the coordinates of L computed earlier and an arbitrary focal length to reconstruct the translation matrix T. Use

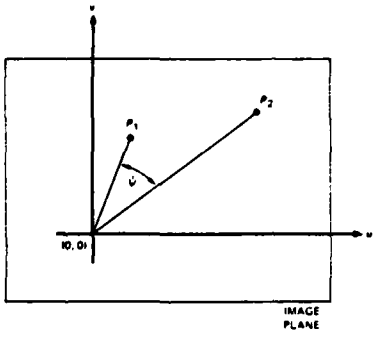


FIGURE 7 DETERMINING THE ROTATION WITHIN THE IMAGE PLANE

the values of θ and ϕ to reconstruct R_x and R_y and use the chosen focal length to construct a perspective projection matrix P'. This comprises a model $M_1 = TR_xR_yP'$, which can be employed to compute the transformation of an arbitrary world point. Call the resulting point $\mathbf{p}_1$.

Next, use the previously determined piercing point to reconstruct the matrix C. Then undo the effects of cropping from the camera model by multiplying the original camera transformation matrix by C^{-1} to obtain

$$M' = MC^{-1} = TRPSCC^{-1} = TRPS.$$

Now the effects of unbalanced scaling will be eliminated. Use the relative scale factor computed earlier to construct a scale transformation matrix:

$$S' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{k_u}{k_v} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then multiply M' by S' to obtain

$$M_2 = M'S' = TRPSS' = TRPS'',$$

where

$$S'' = SS' = \begin{bmatrix} k_u & 0 & 0 & 0 \\ 0 & k_u & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Finally, use M_2 to compute the transformation of the previously chosen world point and call the result $\mathbf{p}_2$.

The angle ψ can now be determined by making use of Equation 1 and the following observation. The only differences between M_1 and M_2 are their focal lengths, a scale factor, and a rotation about the W axis. Although the scale factor is unknown, it is equal in the U and V directions because this was compensated for in computing M_2. Together the scale factor and focal length differences serve only to change the size of the image and impose no other distortions. Observe that $\mathbf{p}_1$ is the image point that would be obtained if there were no rotation about the W axis, no scaling of the image, and no cropping of the image. Similarly, $\mathbf{p}_2$ is the image point that is obtained by starting with the true image point associated with the chosen object point and undoing the effects of cropping and unbalanced scaling. Any difference between $\mathbf{p}_1$ and $\mathbf{p}_2$ must be the result of different-sized images

or of rotation about the W axis. Since this rotation is centered about the origin of this coordinate space, the angle ψ can be determined by measuring the angle between $\mathbf{p}_1$ and $\mathbf{p}_2$ at the origin. The differing focal lengths and scale factors can affect only the distance of the points from the origin and cannot alter the angle between the points when measured from the origin.

4 Discussion

The method presented here provides a straightforward way of determining the parameters of the imaging process from a homogeneous coordinate transformation matrix. The geometric interpretation provides some insight into what the equations mean and when they may fail. The appendix illustrates application of the technique to several sets of real data.

In practice, we must be concerned with the robustness of such an algorithm and how it is affected by errors in the data. For example, if f or k are very large, the view angle subtended by the image is small and the projection is nearly orthogonal. In this case, the method becomes sensitive to the precision of the matrix, and only the camera's orientation can be ascertained with confidence. This property is intrinsic in the problem formulation, and any method that derives camera parameters from the correspondence between image and world coordinates is subject to this sensitivity. The parameters computed by the methods outlined in this paper can be used to reconstruct the camera transformation matrix (within a choice of focal length) when synthetic data are used. When empirical data are used, as in the appendix, instabilities in the matrix often make it impossible to reconstruct it with accuracy.

The camera model used throughout this paper is somewhat simplified. The image plane has been assumed to be perpendicular to the principal ray and the image axes are assumed to be perpendicular. Furthermore, the effects of a non-ideal lens and other nonisotropic distortions have been ignored. The accuracy of the decomposition will degrade if these assumptions are not valid. While we do not expect this technique to be any more robust than that of Ganapathy, we do feel that its geometric interpretation provides useful clues as to when it will be dependable. The method's utility has been demonstrated on actual photographs and because it is non-iterative, the computational burden is insignificant.

5 Acknowledgments

I would like to thank Robert Bolles for his comments and patience in debugging the software for least-squares determination of a camera transformation matrix, and for supplying that program in the first place. I am also grateful for discussions with Kicha Ganapathy as to the relation between this method and his own.

Bibliography

[1] Ballard, D. H., and Brown, C. M., *Computer Vision*, Prentice-Hall, New Jersey, 1982.

[2] Cameron, R., *Above San Francisco*, Cameron and Company, San Francisco, 1976.

[3] Fischler, M. A., and Bolles, R. C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Communications of the ACM, Vol. 24, No. 6, June 1981, pp. 381-395.

[4] Ganapathy, S., "Decomposition of Transformation Matrices for Robot Vision", IEEE, 1984, pp. 130-139.

[5] Gennery, D. B., "Stereo-Camera Calibration", Proceedings of the Image Understanding Workshop, November 1979, pp. 101-107.

[6] Lowe, D. G., "Solving for the Parameters of Object Models from Image Descriptions", IU Workshop, April 1980, pp. 121-127.

[7] Roberts, L. G., "Machine Perception of Three-Dimensional Solids", MIT Lincoln Lab, Technical Report No. 315, May 1963.

[8] Rogers, D. F., and Adams, J. A., *Mathematical Elements for Computer Graphics*, McGraw-Hill Book Company, New York, 1976.

[9] Sobel, I., "On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes", *Artificial Intelligence* 5, 1974, pp. 185-198.

[10] Sutherland, I. E., "Three Dimensional Data Input by Tablet", Proceedings of the IEEE, Vol. 62, No. 4, April 1974, pp. 453-461.

[11] Thompson, Morris M., *Manual of Photogrammetry*, American Society of Photogrammetry, Falls Church, Virginia, 1966.

[12] Ullman, S., *The Interpretation of Visual Motion*, The MIT Press, Cambridge, Massachusetts, 1979.

A Examples

We now present two examples to illustrate our technique.

A.1 Imagery from Robotics Applications

Ganapathy used the following experimentally determined 3 x 4 matrix to demonstrate his method [4].

$$\begin{bmatrix} -2.3819 & 0.49648 & -.039462 & 847.40 \\ -.043897 & -.062872 & -2.4071 & 882.91 \\ -.00026388 & -.00062759 & -.000071843 & 1.0 \end{bmatrix}$$

This matrix is used to obtain the image coordinates (us, vs, s) by premultiplying the world coordinates, (xt, yt, zt, t), by the matrix. To make it compatible with the notation used throughout this paper, it must be transposed and an arbitrary column vector inserted. This column is the one that determines w and does not affect the imaging process. The matrix, suitably rewritten, is

$$M_1 = \begin{bmatrix} -2.3819 & -.043897 & 0.0 & -.00026388 \\ 0.49648 & -.062872 & 0.0 & -.00062759 \\ -.039462 & -2.4071 & 1.0 & -.000071843 \\ 847.40 & 882.91 & 0.0 & 1.0 \end{bmatrix}$$

From this matrix the following values were obtained by the method presented in Section 3.

$$L = (620.51, 1295.68, 321.64)$$

This agrees closely with Ganapathy's determination of the camera's location:

$$(620.9344, 1295.476, 321.8140)$$

The orientation of the camera is computed from the normal to the image plane:

$$\vec{n} = (-.3855, -.9167, -.1049)$$

This yields (in degrees)

$$\theta = 157.1949 \quad \phi = -6.0239 \quad \psi = 359.909$$

in agreement with Ganapathy's results:

$$\theta = 157.1951 \quad \phi = -6.023912 \quad \psi = 359.6915$$

Other parameters obtained from M_1 include

piercing point: $(u_0, v_0) = (682, 478)$ in pixels

focal length: $f \cdot k_u = 3488$

$$f \cdot k_v = 3485$$

relative scale factor: $k_u/k_v = 1.0009$

A.2 Outdoor Imagery

Figure 8 shows a photograph taken from a book of pictures of San Francisco [2]. It was necessary to determine the imaging geometry in order to use the picture for work in image understanding. Ground truth data were obtained manually from a map of the city. A total of fifteen pairs of image and world coordinates were used to obtain the following camera transformation matrix with a least-squares program:

$$M_2 = \begin{bmatrix} .172137 & .131132 & 0.0 & .000346452 \\ -.15879 & .112747 & 0.0 & .000311253 \\ .0187902 & .291491 & 1.27976 & .0000656643 \\ 274.943 & 258.686 & 0.0 & 1.0 \end{bmatrix}$$

The results computed from this image are plotted on the map in Figure 9 and described further below. The camera location was computed to be near the intersection of California and Mason Streets at an elevation of 435 feet above sea level. The camera was oriented as shown on the map, at an angle of 8° above the horizon. Computation of the piercing point is sensitive to errors in the matrix because the projection is nearly orthographic, but the location derived is marked by the point P in the image in Figure 8. The focal length and scale factor relations were computed to be $f \cdot k_u = 495$ and $f \cdot k_v = 560$, indicating an aspect ratio of $k_u/k_v = .88$.

Figures 10 and 11 show the results for another photograph of San Francisco. The camera transformation matrix computed from 16 points of ground truth data is shown here:

$$M_3 = \begin{bmatrix} -.175451 & .0269801 & 0.0 & .000151628 \\ -.105205 & -.0963531 & 0.0 & -.00016085 \\ .0013556 & .23031 & 1.07834 & .0000159749 \\ 297.836 & 249.574 & 0.0 & 1.0 \end{bmatrix}$$

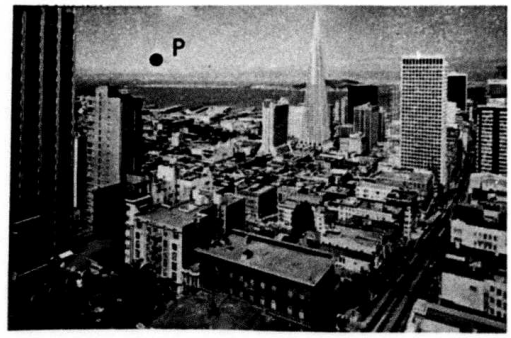


FIGURE 8 PHOTOGRAPH OF SAN FRANCISCO

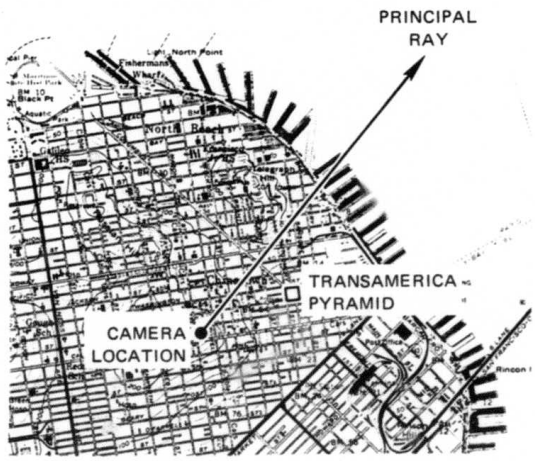


FIGURE 9 MAP OF SAN FRANCISCO

The elevation of the camera was found to be 1200 feet above sea level and the inclination was 4° above the horizon. The horizontal location and orientation are plotted in Figure 11. The piercing point was unreliably computed to be at the point P in Figure 10. The focal length and scale factor relations were $f \cdot k_u = 876$ and $f \cdot k_v = 999$, implying an aspect ratio of $k_u/k_v = .88$.

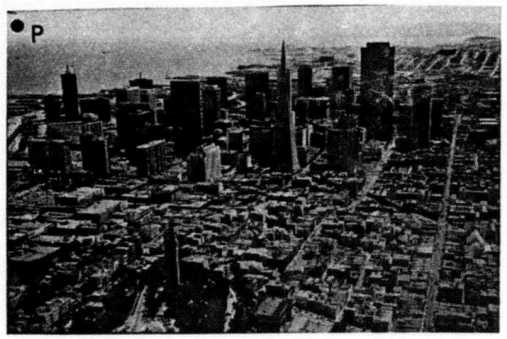


FIGURE 10 ANOTHER PHOTOGRAPH OF SAN FRANCISCO

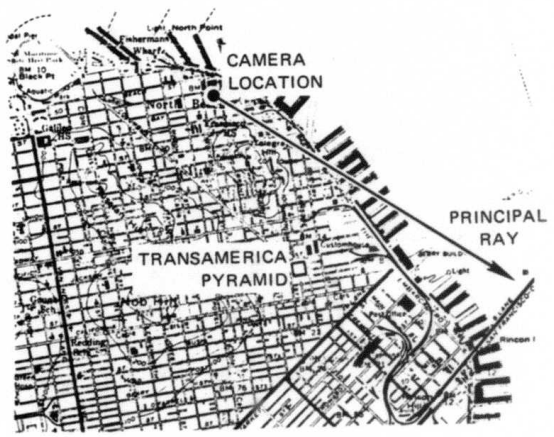


FIGURE 11 MAP OF SAN FRANCISCO

APPENDIX D

Description of SRI's Baseline Stereo System

By: Marsha Jo Hannah

Description of SRI's Baseline Stereo System

Marsha Jo Hannah
Artificial Intelligence Center, SRI International
333 Ravenswood Ave, Menlo Park, CA 94025

Abstract

We are implementing a baseline system for automated area-based stereo compilation. This system, STSYS, operates in several passes over the data, during which it iteratively builds, checks, and refines its model of the 3-dimensional world, as represented by a pair of images. In this paper, we describe the components of STSYS and give examples of the results it produces. We find that these results agree reasonably well with those produced on the interactive DIMP system at ETL, the best available benchmark.

Introduction

Automatic techniques for the production of 3-dimensional data via stereo compilation are receiving increased interest for a variety of applications, including cartography [Panton, 1978], autonomous vehicle navigation [Hannah, 1980], and industrial automation [Nishihara & Poggio, 1983]. Conventional stereo compilation techniques, which are based on area correlation, can produce incorrect results under a variety of conditions, for example, when views are widely separated in space or time, in the vicinity of partial occlusions, in featureless or noisy areas, and in the presence of repeated patterns.

We are investigating ways to overcome these inadequacies. Our research strategy is first to implement a baseline system that performs conventional stereo compilation, then to replace pieces of the system with improved modules as we develop them. Thus, our baseline system will be the core of an ever-improving stereo system. We also intend to test the baseline system against a "challenge data base" of image areas where conventional stereo techniques fail.

As currently implemented, our system includes routines to perform the following operations automatically:

* Select "interesting" points for sparse matching
* Search 2D regions for sparse matches
* If necessary for uncalibrated imagery, compute relative camera parameters from sparse matches
* Compute epipolar lines
* Locate epipolar matches, using disparity estimates from sparse matches when available
* Evaluate matched points for local consistency and believability
* Interpolate between matched points
* Display images and results in left-right stereo, red-green stereo, or as a monocular disparity field
* Compute range data and x-y-z coordinates for matched point pairs
* Display terrain data in perspective with hidden lines removed.

We are currently exploring improved techniques for image matching, for match evaluation, and for terrain surface interpolation.

The Stereo System

Over the past several months, SRI has integrated existing pieces of stereo code into a baseline system for automated area-based stereo compilation. The system operates in several passes over the data, during which it iteratively builds, checks, and refines its model of the 3-dimensional world represented by a pair of images.

The driving program is called STSYS (STereo SYStem). It invokes a variety of modules to perform the necessary processing for stereo compilation. In theory, the modules are independent and can be replaced with improved versions at will; in practice, there are some unavoidable interdependencies of global variables that will have to be attended to.

The following sections describe the components of STSYS in the order they are normally invoked; examples of their results are included. Comments are also made as to improvements that could be made to each of the modules.

REDUCE

The basis for the image matching techniques is a hierarchy of images, as shown in Figure 1. REDUCE is the module that forms this hierarchy from the original images. In the example used for the figures, the original images are a pair of image "chips" digitized from standard 9"x9" mapping photos taken over Phoenix South Mountain Park, near Guadalupe (a suburb of Phoenix), Arizona. These images are 2048x2048 pixels in size, and cover an area that is approximately 2 kilometers square on the ground; elevations in the area range from 360 to 540 meters. The reduction hierarchy consists of a pyramid of images, each at half of the resolution of its parent; in this case REDUCE produces pairs of images that are 1024x1024, 512x512, 256x256, 128x128, 64x64, 32x32, and 16x16 pixels in size. (Figure 1 shows only the 256x256 through 16x16 image pairs.)

At present, REDUCE produces pixels in each reduced image by simple averaging of the pixels in an NxN square in the next-largest image (in the above case, N=2). It is known that this technique can produce artifacts in the data, and a more sophisticated technique of convolving the image with a Gaussian, then sub-sampling, is preferred [Burt, 1980]. Substitution of this technique will be one of the first enhancements made to STSYS.

INTEREST

The first step in the matching process is to procure a set of well-scattered, reliable matches in the image. Our approach is first to select areas in one image that contain sufficient information to produce reliable matches. To accomplish this, a statistical operator based on image variance and edge directionality is passed over the image; local peaks in the output of this operator are recorded as the preferred places to attempt the matching process.

Historically, such operators have been called *interest* operators, and the peaks in the operator output have been called *interesting points* [Moravec, 1980]. This nomenclature is somewhat misleading, as the points selected are rarely interesting to a human observer; however, these terms have been in use in the computer vision community for over 10 years. It should be noted that present interest operators are not feature detectors; the same operator run over both images of a stereo pair will not necessarily pick out the same points in the two images. In our system, the interest operator is run in only one of the images, where it selects points that are to be matched in the second image by other means. (A possible enhancement to STSYS would be to design and implement efficient interest operators that really do choose "interesting points," such as crossroads, building corners, sharp bends in rivers, etc.)

INTEREST permits the user to specify the operator to be used [Hannah, 1980], the window size over which it is calculated, and the window radius for testing local peaks. It also provides the capability to divide the image into a grid of subimages, and records the relative ranks of the interesting points within their grid cells; this permits the most interesting point(s) for each area to be matched first. Figure 2 shows the interesting points for the right image of the Phoenix pair; the numbers indicate the 1st, 2nd, 3rd, and 4th most interesting points in a 6x6 grid of cells.

Preliminary Matching

At this point in the processing, it is possible to take one of two different approaches to the matching. If nothing is known regarding the absolute camera positions and orientations (as would be the case for an amateur, handheld stereo pair), an unstructured hierarchical matching algorithm is used on the most interesting points. The results of these matches are used in seeking a solution for a simplistic relative camera model (5 angles describing the relative positions and orientations of 2 ideal cameras [Hannah, 1974]), which can then be used for the epipolar constraint in further matching. This approach uses the modules HMATCH and C2MODEL, described below. On the other hand, if the camera parameters are known (as would be the case for the highly calibrated cartographic stereo images intended for terrain mapping) matching can proceed directly with the epipolar constraints, using the module LMATCH.

HMATCH

HMATCH assumes that nothing is known about the relative orientations of the images, other than that they cover approximately the same area, at about the same scale, with no major rotation between the images. It matches each specified point (usually the most interesting point in each grid cell) using an unguided hierarchical matching technique similar to that reported in [Moravec, 1980]. This technique begins with the point in the largest image (the 2048x2048 right image of the Phoenix set), traces it back through that image's hierarchy (in our example, it repeatedly halves the co-ordinates of the point) until it reaches an image that is approximately the size of the correlation window (the 16x16 image for the 11x11 correlation windows that we used). It then uses a 2-dimensional spiral search, followed by a hill-climbing search for the maximum of the normalized cross-correlation between the image windows [Quam, 1971]. This global match is then refined back down the image hierarchy; that is, the disparity at each level (suitably magnified to account for relative image scales) is used as a starting point for a hill-climb search at the next level. The correlation window size remains constant at all levels of the hierarchy, so the match is effectively performed first over the entire image, then over increasingly local areas of the image. This technique permits the use of the overall image structure to set the context for a match; the gradually increasing detail in the imagery is then followed down through the hierarchy to the final match.

Figure 3 shows the results of this technique on a point in the Phoenix set. The image hierarchy is the same as in Figure 1, with the addition of 63x63 image chips covering the matched area in the 2048x2048, 1024x1024, and 512x512 images; these are shown in the upper right corner of each hierarchy. The matching began in the right image in the 2048x2048 chip, traced the right point through the hierarchy (approximately clockwise in the figure) to the 16x16 right image, matched it to the 16x16 left image, then refined it back through the left image hierarchy until reaching the left 2048x2048 chip.

It is instructive to look at the correlation coefficients for these matches (see Table 1). In the smaller images, the correlation is poor, since the window covers a large area of terrain with a great deal of relief. As the matching moves up the hierarchy, the correlation improves, because the window now approximates an area at a single elevation. After reaching the 512x512 images, however, the correlation begins to decline, both in absolute value and with respect to an autocorrelation-based threshold [Hannah, 1974]. This is due to noise in the images; if one examines the chip from the 2048x2048 left image, one will see several streaks across the image, representing scratches on the original photograph and/or dropped data in the digitization; close examination also reveals a grainy noise pattern. Because the degraded correlations will cause difficulties in determining which matches are the correct ones, our processing has gone only to the 512x512 images. More code will be added to STSYS in order to refine the final matches from this level down to the original 2048x2048 images.

Figure 4 shows the results of HMATCH on the most interesting point in each grid cell. Only the points thought to have been matched correctly are shown; those with poor correlation or whose matches fell outside of the image have been discarded by STSYS.

C2MODEL

If no camera calibration information is available, the module C2MODEL calculates a simplistic relative camera model from a set of matched point pairs. This is accomplished by searching for 5 angles--the azimuth and elevation of the second camera's focal point with respect to the first camera; and pan, tilt, and roll of the second camera's axes with respect to those of the first. The object of the search is to minimize the error between the matched point in the second image and the epipolar line produced when the point in the first image is projected through the hypothesized cameras. The search proceeds by a linearization of the equations and their analytic derivatives [Gennery, 1980]. Once a solution is found, the reliability of the matched points is assessed. Points that appear to contribute too much error to the solution are removed from the calculation, and the solution is redone. Either this process reaches a successful conclusion when the point set is found to be consistent, or it reports failure if too many of the point pairs are rejected.

The resulting camera model is quite crude, as it must depend on a guess as to the focal lengths of the cameras and the length of the baseline between the cameras. Also, it assumes that we are using ideal cameras, thus totally ignoring the internal calibration of the cameras. It is, however, suitable for approximating the epipolar constraint to simplify further matching.

LMATCH

If the camera parameters are given (or once the crude ones have been derived), matching can proceed somewhat more efficiently. The camera parameters define the manner in which a point in the first image projects to a line in the second image--the epipolar constraint. This constraint can be used to cut the search from 2 dimensions (all over the image) to 1 dimension (back and forth along the epipolar line).

LMATCH proceeds very much like HMATCH, except that the search for a match is confined to the vicinity of the epipolar line. Because we assume that there is no outside information to indicate where these preliminary matches lie along the line, we again use the hierarchical technique to search out and refine the match. If relative camera parameters have been derived, LMATCH is used on the second most interesting point per grid cell, plus any points that C2MODEL indicated were unreliable; the results of this mode are shown in Figure 5. If the true camera parameters have been supplied, LMATCH is used on the two most interesting points in each grid cell; these results are shown in Figure 6.

Anchored Matching

Once several reliable matches have been found, they can be used as "anchor" points for further matching. Our basic technique for this again uses the grid cells in the image. A given point will lie in some grid cell; the closest matched point(s) will lie in that cell or in one of the 8 neighboring cells. Under the assumption that the world is generally continuous, a point would be expected to have a disparity similar to that of its nearest neighbors. Thus, to approximate the disparity at a point, we first calculate the average of the disparities of the well-matched points in the current and neighboring cells, weighted by the inverse of the distance between the current point and the neighboring point. (As we develop more sophisticated interpolation schemes, this disparity approximation technique will be upgraded.) This approximate disparity is used along with the epipolar constraint to perform a very local search for the match to a point. Note that a point is considered to be well-matched if it has a correlation above a user-settable absolute threshold, usually 0.5, as well as having a correlation above a variable threshold, based on the autocorrelation function around the point in the first image (see Table 1 for examples). Our definition of well-matched should also be upgraded to include distance off of epipolar lines as well as a measure of how consistent the disparity is with its neighbors.

PMATCH

At this point in our processing, we have matched the two most interesting points in each grid cell. This is still rather sparse information, so we next invoke the module PMATCH to match the balance of the interesting points. It uses the anchored match technique described above, with a generous search radius along the epipolar line, to find these matches. Figure 7 shows the results of this module. Two different marks are used for the matches, denoting whether their correlations indicate that they are well-matched.

GMATCH

We next produce matched points on a closely spaced grid. The module GMATCH also uses the anchored match technique, with a somewhat restricted search radius along the epipolar line, to calculate matches on a user-specified grid. Figure 8 shows the results of this module on a 20x20 grid, again using different marks for the different qualities of match.

Terrain Modeling

Given the dense grid of matched points and the camera calibration, it is possible to derive a digital terrain model. If external and internal camera information is available, the module SRIDTM can be used to create a reasonably accurate DTM, which can then be displayed with another program, DTMICP. (An example of DTMICP output is shown in Figure 9; it can also produce range images of the terrain or pictures of the original imagery "painted" on the terrain.) If the only camera information is C2MODEL's relative model, then the module RELDEPTH can be used to create a relative DTM. However, due to the many over-simplifications and the computational instability of the relative camera model, such relative DTMs are of very low accuracy, and their use is discouraged.

Often, a terrain model is desired that has its points more closely spaced than that provided by the stereo matching process. Sometimes, there may be areas of the images that cannot be matched, due to noise in the data, insufficient information, or changes such as moving vehicles; this will result in "holes" in the grid of terrain data, which must be filled in somehow. In either case, interpolation of the matched data points is necessary to provide information at other points. Work on this topic is reported separately [Smith, 1984].

Evaluation

Evaluation of the accuracy of STSYS is difficult, as there do not seem to exist stereo data sets with known ground truth to compare against our results. We do, however, have the results of an interactive stereo compilation algorithm called Digital Interactive Mapping Program (DIMP), produced and operated by the U.S. Army Engineer Topographic Laboratories (ETL) [Norvelle, 1981]. It should be noted, however, that ETL's results were obtained by an interactively coached process, which was run on a 5x5 grid in the 2048x2048 images and used correlation windows warped to account for the local steepness of the terrain, while ours were obtained by a fully automatic process that ran on a 20x20 grid in the 512x512 images without warping; comparing them is a little like comparing apples and oranges. (Another planned upgrade to our matching techniques is the use of warped correlation in the match refining step.)

Of our matches (both interesting points and grid points), approximately 98% agree reasonably well with the nearby ETL matches at the resolution of the 512x512 images. Of the remaining 2%, most are clearly blunders on our part, although a few appear to be the result of errors in the DIMP compilation. It is not known what fraction (if any) of the 98% represent places where our processing and the DIMP processing produced similar wrong answers.

Discussion

SRI has an operational baseline system for automated area-based stereo compilation. This system, STSYS, operates in several passes over the data, during which it iteratively builds, checks, and refines its model of the 3-dimensional world represented by a pair of images. In this

paper, we have described the components of STSYS and given examples of the results it produces. We have compared these results to those produced on the interactive DIMP system at ETL, and found that they compare favorably.

STSYS is, at present, an experimental program; no attempt has been made to optimize it for best results or fastest operation. The program is still evolving, and will not be ready for transfer to other users until its methods stabilize. Likewise, more complete documentation must wait on completion of the code.

Acknowledgements

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract MDA903-83-C-0027, which is monitored by the U.S. Army Engineer Topographic Laboratory. The views and conclusions contained in this paper are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government.

I would like to thank Robert Bolles, Lynn Quam, Grahame Smith, and Martin Fischler for their support on this project.

References

Burt, Peter J., 1980. "Fast, Hierarchical Correlations with Gaussian-like Kernels," University of Maryland Computer Science Center Report TR-860, January, 1980.

Gennery, Donald B., 1980. "Modelling the Environment of an Exploring Vehicle by means of Stereo Vision," Ph.D. Thesis, Stanford University Computer Science Department Report STAN-CS-80-805, June, 1980.

Hannah, Marsha Jo, 1974. "Computer Matching of Areas in Stereo Images," Ph.D. Thesis, Stanford University Computer Science Department Report STAN-CS-74-438, July, 1974.

Hannah, Marsha Jo, 1980. "Bootstrap Stereo," *Proceedings: Image Understanding Workshop,* College Park, MD, April, 1980, pp. 201-208.

Moravec, Hans P., 1980. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Ph.D. Thesis, Stanford University Computer Science Department Report STAN-CS-80-813, September, 1980.

Nishihara, H. Keith, and Tomaso Poggio, 1983. "Stereo Vision for Robotics," *Proceedings of the International Symposium of Robotics Research,* Bretton Woods, NH, September, 1983.

Norvelle, F. Raye, 1981. "Interactive Digital Correlation Techniques for Automatic Compilation of Elevation Data," U.S. Army Engineer Topographic Laboratories Report ETL-0272, October, 1981.

Panton, Dale J., 1978. "A Flexible Approach to Digital Stereo Mapping," *Photogrammetric Engineering and Remote Sensing,* Vol. 44, No. 12, pp. 1499-1512.

Quam, Lynn H., 1971. "Computer Comparison of Pictures," Ph.D. Thesis, Stanford University Computer Science Department Report STAN-CS-71-219, May, 1971.

Smith, Grahame B., 1984. "A Fast Surface Interpolation Technique," SRI International Artificial Intelligence Center Technical Memo, in preparation, June, 1984.

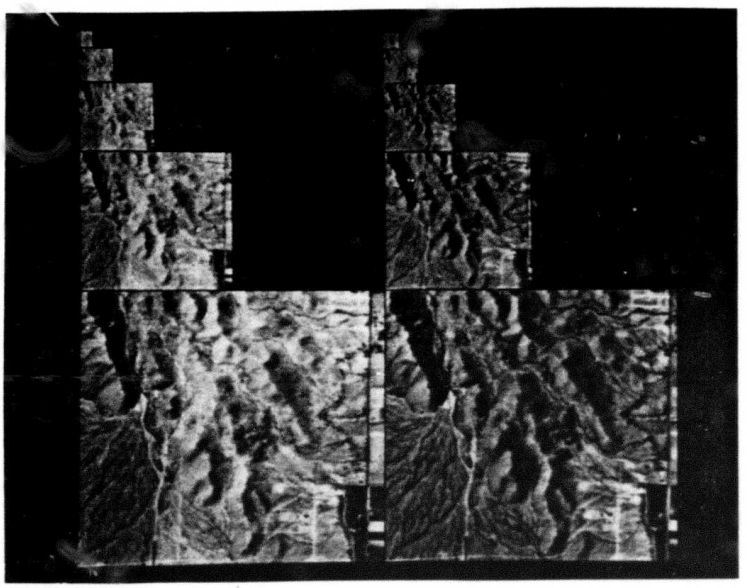


Figure 1--Reduction Image Hierarchy.

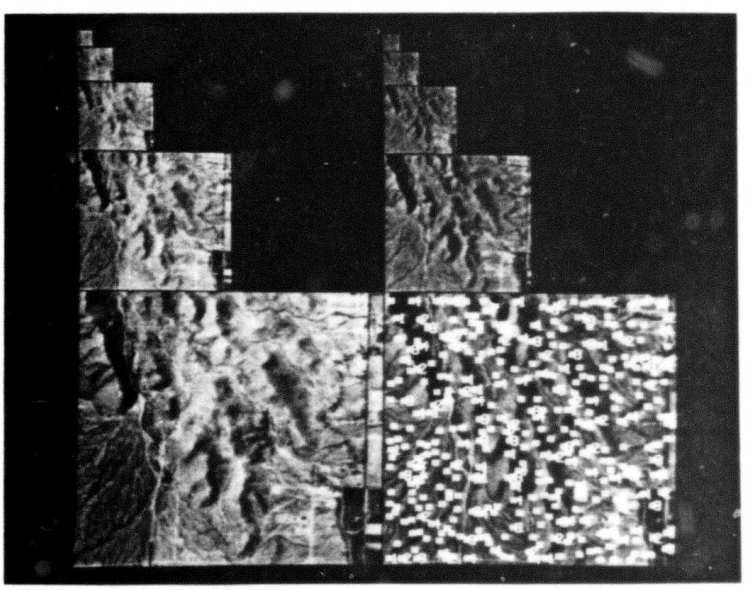


Figure 2--Interesting Points, Ranked by Grid Cell.

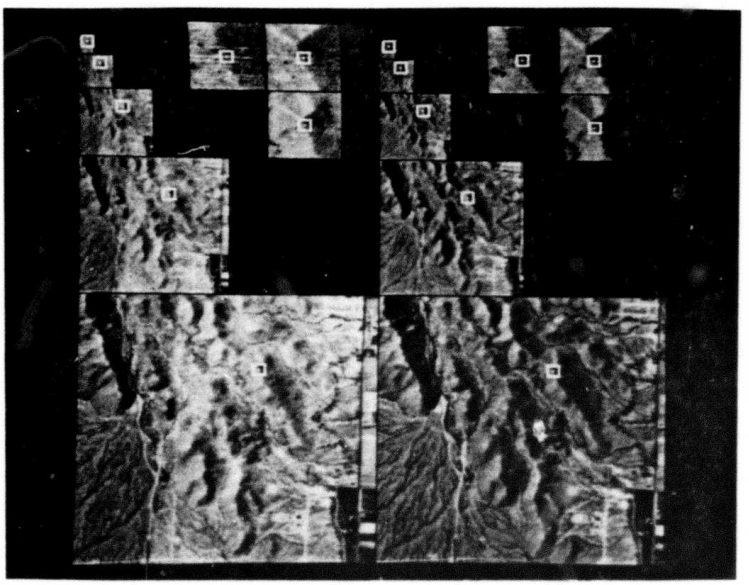


Figure 3--Hierarchical Match of an Interesting Point.

Table 1--Hierarchical Correlations for Point in Figure 3.

Image size	Point 1	Point 2	Correlation	Autocorrelation
16x16	(9,11)	(9,11)	0.140452	0.577717
32x32	(19,23)	(19,23)	0.384883	0.437053
64x64	(38,46)	(37,46)	0.738581	0.738427
128x128	(77,92)	(76,92)	0.929933	0.885289
256x256	(154,184)	(153,184)	0.954606	0.918228
512x512	(308,369)	(306,369)	0.916062	0.929428
1024x1024	(616,738)	(612,737)	0.750448	0.932947
2048x2048	(1232,1476)	(1222,1475)	0.341622	0.790917

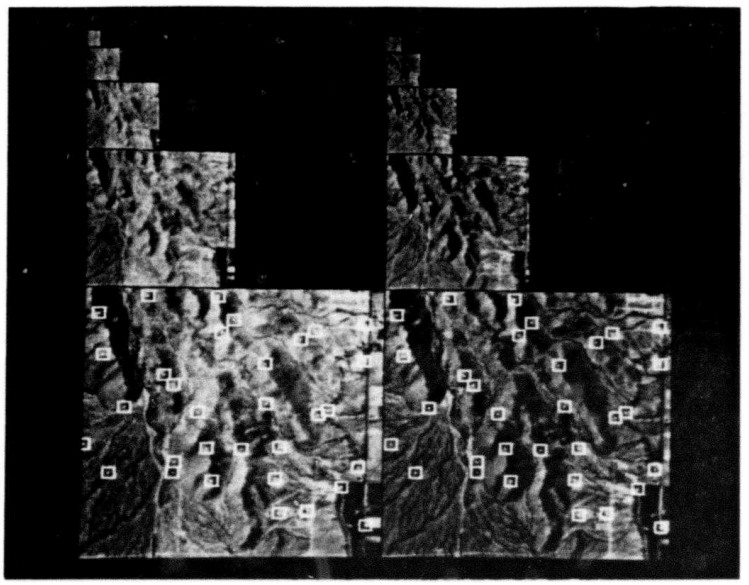


Figure 4--Results of Unstructured Hierarchical Matching of Most Interesting Point in Each Grid Cell.

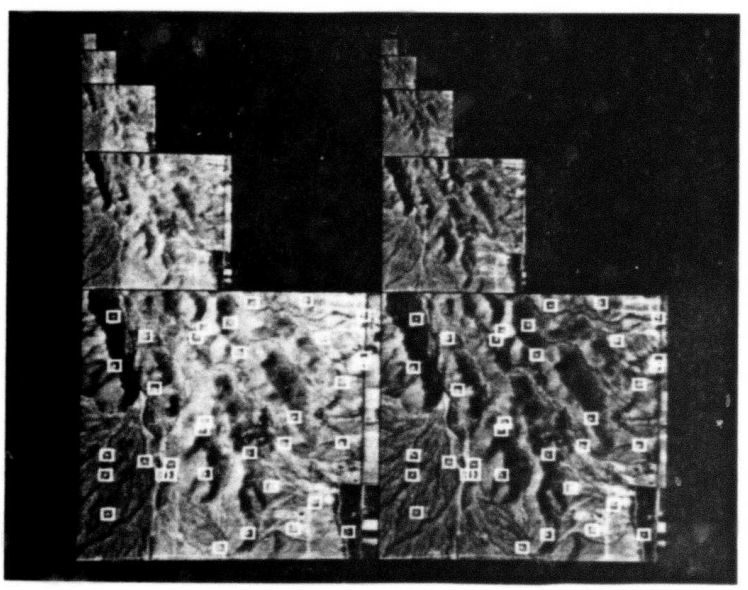


Figure 5--Results of Epipolar Hierarchical Matching of Second Most Interesting Point in Each Grid Cell.

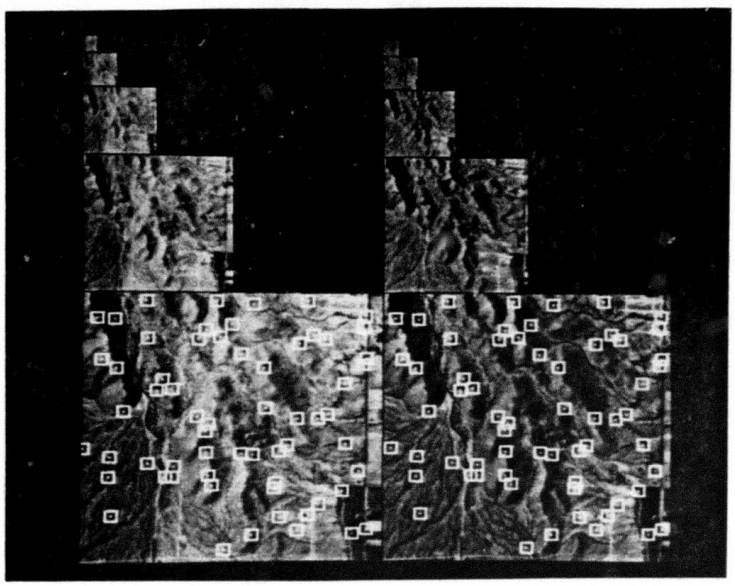


Figure 6--Results of Epipolar Hierarchical Matching of Two Most Interesting Point in Each Grid Cell.

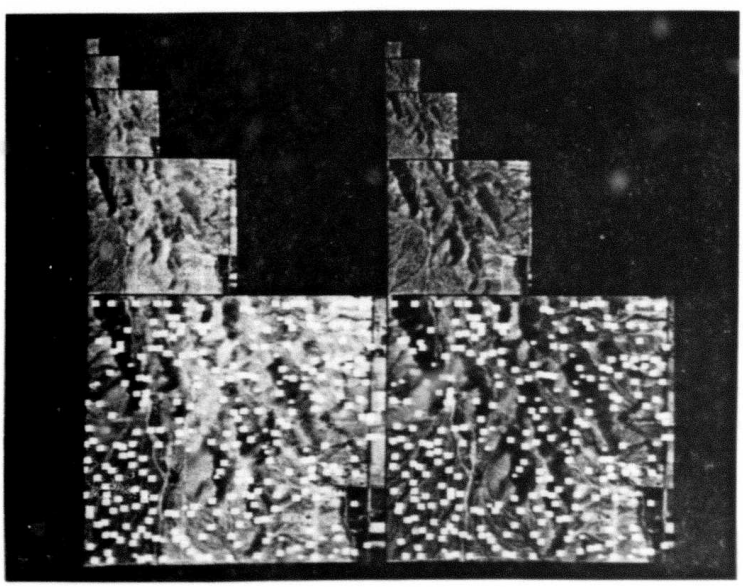


Figure 7--Results of Anchored Epipolar Matching of Remaining Interesting Points.

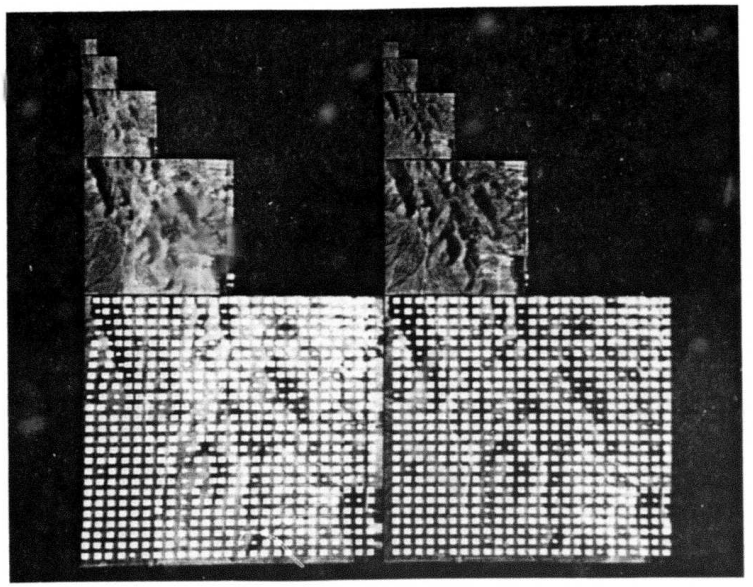


Figure 8--Results of Anchored Epipolar Matching of a Grid of Points.

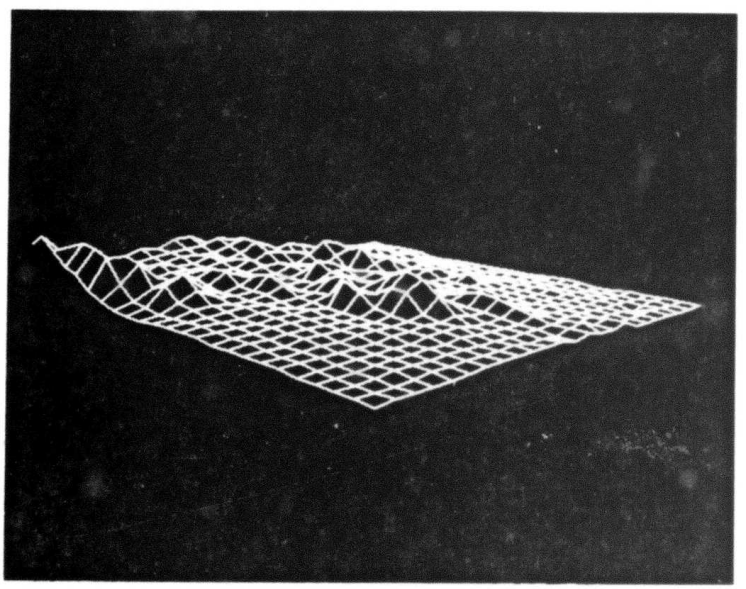


Figure 9--A View of the Resulting Digital Terrain Model.

APPENDIX E

A Fast Surface Interpolation Technique
By: Grahame B. Smith

A Fast Surface Interpolation Technique.

Grahame B. Smith

Artificial Intelligence Center, SRI International
Menlo Park, California 94025

Abstract

A method for interpolating a surface through 3-D data is presented. The method is computationally efficient and general enough to allow the construction of surfaces with either smooth or rough texture.

1. Introduction

In image analysis we are often faced with the fact that the measurements we make in an image only constrain properties of the 3-D world, instead of specifying them. Analysis techniques that recover 3-D shape information from image measurements incorporate very restrictive assumptions about the nature of the world. In our attempts to avoid the need for these restrictions, we have been examining hypothesis-and-test methods. If we assume that we are able to obtain some shape data, from which we can hypothesize an approximate shape model for the world, then we can use this model to predict image features. To proceed from shape data to an approximate shape model we need to "flesh out" the data. In this paper we address the problem of fitting a surface to a set of points whose 3-D locations are known. While our interest centers on fitting a surface to 3-D location data that have been acquired by processing images of that surface, the technique developed has application to a broad class of surface-fitting tasks.

To select a surface-fitting procedure, it is insufficient merely to know the data set and to require that a surface be fitted to the points in that set. We also need to know the desired properties of the surface, the characteristics of the data, and the uses to which the fitted surface will be put. If we are building a surface to allow, say, water runoff estimates to be made, smoothness may be a desired property for that surface. For realistic rendering of a natural surface in computer graphics, however, a fractal surface may be preferable. While the technique we develop can construct either smooth or rough surfaces, our applications generally require the former. Our examples, Figures 3 and 4, show both types.

Besides the desired properties of the fitted surface, the characteristics of the data limit the approach we must adopt to surface construction. In fitting a surface we must balance the influence exerted by the data values themselves, against that exerted by the implicit surface model embedded in any fitting procedure. If our data values are inaccurate and we know the class of surfaces that should fit the data, we can usually let the surface model dominate the construction process. Least-square methods are typical of procedures that prefer a model to data. In general, techniques whose resultant surfaces do not conform exactly to the data are known as approximation methods. Methods that produce surfaces conforming exactly to the data are called interpolation methods.

The selection of an approximation or interpolation method is influenced by properties of the data other than their accuracy. Consider, for example, the terrain data collected by a surveyor. In selecting the places at which to make measurements, he considers the breakpoints of the surface – that is, those places on the surface where the gradient is discontinuous – and his data include measurements at these breakpoints. Surface reconstruction by linear interpolation over triangular surface patches is possible because the surveyor has furnished not only the 3-D data, but also an implicit statement that the surface between his points can be approximated by planar patches. In matching stereo pairs of images, an edge-based matcher provides more than the 3-D data it produces. Like a surveyor's data, it too makes an implicit statement about the continuity of the imaged surfaces. On the other hand, an area-based correlation matcher says less about surface continuity, but has the desirable behavior of providing regularly spaced data.

Such data can usually be processed with considerably less computational effort than data that are irregularly spaced. The volume of data, the regularity of their spacing, the implicit characteristics of their collection procedure, and their accuracy are all essential parameters in selecting a surface-fitting technique. For our applications we choose to investigate interpolation methods. We want methods that will work with irregularly spaced data, but still achieve substantial computational savings if we can use a regular grid of data points. We need to be able to handle thousands of such points. As a rule, we do not want to use implicit properties of the data that stem from their collection procedure.

The uses to which the fitted surface will be put further restricts the set of applicable surface-fitting procedures. If the task at hand is surface area estimation, the accuracy of the surface gradients is not important. Conversely, if we wish to use the fitted surface to generate the latter's image under some known lighting conditions, the surface gradient information then becomes crucial. We can classify the uses of fitted surfaces by the surface derivatives that are needed. An application that does

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract MDA903-83-C-0027 and by the National Aeronautics and Space Administration under Contract NASA 9-16664. These contracts are monitored by the U.S. Army Engineer Topographic Laboratory and by the Texas A&M Research Foundation for the Lyndon B. Johnson Space Center.

not require surface derivatives to be calculated can usually be satisified by a surface composed of local patches. That is, the surface is fitted locally patch by patch, with each patch determined by a small number of local data points. Such methods have strong surface models and few data are needed to instantiate them. As a consequence, however, the surface derivatives are more a function of the surface model than of the data. The amount of data used to determine the surface patch may be barely sufficient to calculate an average value for the surface derivatives across the whole patch; besides, any variations in derivatives across the patch are caused by the model, not the data. The more data employed, the less is the influence of the surface model on the calculation of surface derivatives. In the extreme case, all the data may be used to determine the surface to be fitted at each locality. Such techniques are called global methods, whereas those that use only local data are local methods. Our applications require that we calculate surface curvature from our fitted surface. The technique we present here is a global method for surface fitting.

In summary, we address the problem of fitting a surface to a large data set composed mostly of regularly spaced data points, but which also includes grid points at which we have no data, and non grid points where data values are known. The data are acquired through a collection process that is assumed to yield accurate values, but for which we choose not to characterize the data further. We require a solution that is smooth and from which we can calculate the first and second surface derivatives. We present details of a global interpolation method that is computationally efficient and appears to applicable to a broad range of tasks. Although the general form of the method applies to non gridded data our computationally efficient algorithm comes from exploiting the regularity of the data points.

We commence by considering the multiquadric method invented by Hardy [1] for modeling natural terrain. In its generalized form, we examine it under the restriction of regularly spaced data points and derive an algorithm to solve for the unknown parameters. We show how to generate the interpolated surface in an efficient manner.

2. Surface Interpolation

2.1 Hyperbolic Multiquadrics

Suppose we have a set of data points, $\{(x_i, y_i, z_i)\}_{i=0}^{n-1}$ in 3-D space to which we wish to fit a hyperbolic multiquadric surface [1] defined by

$$z(x,y) = \sum_{j=0}^{n-1} c_j [d_j^2(x,y) + h]^{\frac{1}{2}} \quad ,$$

where $d_j^2(x,y) = (x - x_j)^2 + (y - y_j)^2$, h is a user-specified constant, and c_j's are the coefficients that must be determined.

To understand this method, let us suppose that $h = 0$. The data are fitted by placing a cone at each of the n data points so that the cone's axis is aligned with the z axis direction and the cone's apex is in the $z = 0$ plane. That is, the data are fitted with a set of cones, some of which are inverted. The z value of the constructed surface at position (x,y) is calculated by summing the z values contributed by each of the n cones at this (x,y) position.

Each cone has one free parameter, namely, its apex angle; we determine these apex angles by requiring that the constructed surface pass exactly through the data points. In the foregoing expression, the c_j's correspond to the apex angles of the cones. We calculate the c_j's by solving the nxn system of linear equations

$$\sum_{j=0}^{n-1} c_j [d_j^2(x_i, y_i) + h]^{\frac{1}{2}} = z_i \qquad i = 0, ..., n-1 \quad .$$

Note that this fitting technique does not require that the data be regularly spaced; furthermore, when $h \neq 0$, hyperboloids rather than cones are fitted to the data. Cones and hyperboloids are not the only options. Stead [2], for example, has generalized this method, using the form

$$z(x,y) = \sum_{j=0}^{n-1} c_j [d_j^2(x,y) + h]^{\frac{g}{2}} \quad .$$

2.2 General Form

We examine surface-fitting techniques that use the general form of the above method, namely,

$$z(x,y) = \sum_{j=0}^{n-1} c_j g(x - x_j, y - y_j) \quad ,$$

where the kernel function g is any function of the parameters $x - x_j, y - y_j$. Clearly, the previously defined functions are particular cases of this form. As before, we determine the c_j's by solving the nxn system of linear equations

$$\sum_{j=0}^{n-1} c_j g(x_i - x_j, y_i - y_j) = z_i \qquad i = 0, ..., n-1 \quad .$$

For large values of n it is not feasible to solve this system of equations. In our applications n may be 10,000. However, for smaller n we have used the above form to "patch" holes in a regular grid of data points. While any kernel function can be employed, we have found it important to match the method used to solve the nxn system of linear equations to the form of the kernel function selected. The numerical difficulties encountered in solving some of the systems of equations produced by a particular kernal can often be averted by exploiting properties of the linear system stemming from the choice of kernel function. For example, if we use the Gaussian function as the kernel, the symmetric positive definite coefficient matrix of the system of linear equations allows solution by the "square-root" method (see, for example [3]), and avoids the numerical problems created by Gaussian elimination. If we impose the restriction that the data points must be gridded, we can find feasible solution techniques even when n is of the order of millions.

2.3 Regular Grid Solution

Consider the problem of fitting the surface

$$z(x,y) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{i,j} g(x - x_{i,j}, y - y_{i,j}) \quad , \qquad (1)$$

where $[(x_{i,j}, y_{i,j})]_{i=0,j=0}^{n-1\,m-1}$ is an nxm regular grid, to the data set $[(x_{i,j}, y_{i,j}, z_{i,j})]_{i=0,j=0}^{n-1\,m-1}$. We can find an expression for calculating the c_{ij}'s in the following manner.

Let $G(u,v)$ denote the discrete Fourier transform (DFT) of $g(x,y)$. Using the shift theorem of DFT theory, we note that the DFT of $g(x-x_{i,j}, y-y_{i,j})$ is $G(u,v)e^{-2\pi i(\frac{u}{n}x_{i,j}+\frac{v}{m}y_{i,j})}$. If $Z(u,v)$ denotes the DFT of $z(x,y)$, we can write the DFT of Equation (1), namely,

$$Z(u_{k,l}, v_{k,l}) = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} c_{i,j} G(u_{k,l}, v_{k,l}) e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})} \qquad k=0,\ldots,n-1 \quad l=0,\ldots,m-1.$$

Removing $G(u_{k,l}, v_{k,l})$ from the summation, we have

$$\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} c_{i,j} e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})} = \frac{Z(u_{k,l}, v_{k,l})}{G(u_{k,l}, v_{k,l})} \quad .$$

Taking the inverse DFT of the above expression, we obtain

$$\sum_{k=0}^{n-1}\sum_{l=0}^{m-1}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} c_{i,j} \cdot e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})} e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})} = \sum_{k=0}^{n-1}\sum_{l=0}^{m-1} \frac{Z(u_{k,l}, v_{k,l})}{G(u_{k,l}, v_{k,l})} e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})} \quad .$$

Using $\mathbf{F}^{-1}$ to represent the inverse DFT, $\mathbf{F}^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{p,q}, y_{p,q})$ is the inverse DFT of $\frac{Z(u,v)}{G(u,v)}$, calculated at $(x_{p,q}, y_{p,q})$. We have

$$\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} c_{i,j} \sum_{k=0}^{n-1}\sum_{l=0}^{m-1} e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})} \cdot e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})} = \mathbf{F}^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{p,q}, y_{p,q}) \quad .$$

Now if $x_{i,j} = x_{p,q}$ and $y_{i,j} = y_{p,q}$,

$$\sum_{k=0}^{n-1}\sum_{l=0}^{m-1} e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})} e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})} = nm$$

otherwise

$$= 0$$

Hence

$$c_{i,j} = \frac{1}{nm}\mathbf{F}^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{i,j}, y_{i,j}) \quad . \qquad (2)$$

An alternate way of viewing the above derivation is to note that $g(x_{p,q}-x_{i,j}, y_{p,q}-y_{i,j})$ forms a circulant matrix, and to recall that such matrices are diagonalized by the discrete Fourier transform [4].

2.4 Surface Rendering

Once the $c_{i,j}$'s have been calculated, Equation (1) provides an analytic expression for the constructed surface. We can calculate $z(x,y)$ for any (x,y) position. However, each such calculation involves the sum of nm terms. If it is our intention to use this analytic expression for surface interpolation we may have to calculate this sum a very large number of times. The cost savings gained in computing the $c_{i,j}$ coefficients by means of the DFT (implemented by the fast Fourier transform) will be offset by the cost of these summations. As a rule, if we commence with data on a regular grid we want an interpolated surface on a finer grid. This results in considerable savings which are realized when the DFT is again employed.

Suppose we want to interpolate each grid interval in x and y at an additional number of points so that the final surface is calculated on a *rnxsm grid*. Consider Equation (1), revised for the new, larger grid:

$$z(x,y) = \sum_{i=0}^{rn-1}\sum_{j=0}^{sm-1} c'_{i,j} g(x-x_{i,j}, y-y_{i,j}) \quad , \qquad (3)$$

where

$$c'_{i,j} = c_{i+r,j+s} \quad i \pmod r = 0, j \pmod s = 0,$$
$$= 0 \quad \textit{otherwise.}$$

That is, we assume the surface is constructed by the placing of objects at each of the new grid points, but zero coefficients are associated with all objects except those placed at the original data points. Now, taking the DFT of Equation (3), we get

$$Z(u_{k,l}, v_{k,l}) = \sum_{i=0}^{rn-1}\sum_{j=0}^{sm-1} c'_{i,j} G'(u_{k,l}, v_{k,l}) \cdot e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{rn}+\frac{v_{k,l}y_{i,j}}{sm})} \quad \textit{for} \quad k=0,\ldots,rn-1 \quad l=0,\ldots,sm-1,$$

i.e.,

$$Z(u_{k,l}, v_{k,l}) = G'(u_{k,l}, v_{k,l}) C'(u_{k,l}, v_{k,l}) \quad ,$$

where $G'(u,v)$ is the DFT of $g(x,y)$, defined on the finer grid, and $C'(u,v)$ is the DFT of the array $c'_{i,j}$.

The interpolated surface can be formed by taking the inverse DFT of the above expression:

$$z(x_{i,j}, y_{i,j}) = \mathbf{F}^{-1}[G'(u_{k,l}, v_{k,l}) C'(u_{k,l}, v_{k,l})] \quad .$$

Note that, in finding the $c_{i,j}$'s in Equation (2), we took the inverse DFT of $\frac{Z(u,v)}{G(u,v)}$, then stretched these $c_{i,j}$'s by adding zeros at the points corresponding to the new interpolation points, and finally took the DFT of the stretched coefficients to calculate $C'(u,v)$. These steps are in fact unnecessary, for we can calculate the $C'(u,v)$'s directly from the $\frac{Z(u,v)}{G(u,v)}$'s. The similarity theorem of DFT theory [5] is required:

$$C'(u_{k,l}, v_{k,l}) = \frac{1}{rs} \frac{Z(u_{k,l} \pmod n, v_{k,l} \pmod m)}{G(u_{k,l} \pmod n, v_{k,l} \pmod m)} \qquad (4)$$
$$k=0,\ldots,rn-1 \quad l=0,\ldots,sm-1.$$

2.5 Algorithm

We can now write down our interpolation algorithm:

1. Given the data array $z(x_{i,j}, y_{i,j})$, find its DFT, $Z(u_{i,j}, v_{i,j})$
2. Find the DFT, $G(u_{i,j}, v_{i,j})$, of the kernel function, $g(x_{i,j}, y_{i,j})$
3. $C(u_{i,j}, v_{i,j}) = \frac{Z(u_{i,j}, v_{i,j})}{G(u_{i,j}, v_{i,j})}$
4. Calculate $C'(u_{i,j}, v_{i,j})$, using Equation (4)
5. Calculate $G'(u_{i,j}, v_{i,j})$ for the larger interpolation grid
6. $Z'(u_{i,j}, v_{i,j}) = C'(u_{i,j}, v_{i,j})G'(u_{i,j}, v_{i,j})$
7. Find the interpolated surface by taking the inverse DFT of $Z'(u_{i,j}, v_{i,j})$.

Note that for selected kernel functions Steps 2 and 5 could be precomputed for standard-size grids. As an alternative, these steps can sometimes be accomplished by analytic means if the analytic form of the kernel function is known.

3. Discussion

For purposes of illustration, let us compare the computational efficiency of this method on a regular grid with the cost of the usual non gridded formulation of the multiquadric. Of course, since the usual formulation deals with irregularly spaced data, we would not expect it to compare favorably with this method; such a comparison nevertheless confirms the advantages of our technique. Consider a square nxn grid of data points on which we want an interpolated surface over a $rnxrn$ grid. The usual multiquadric formulation solves a n^2xn^2 system of linear equation at a cost proportional to n^6, and calculates $rnxrn$ sums of terms at a cost proportional to r^2n^4. If it is assumed that $n > r$, this cost is dominated by the n^6 term.

The algorithm outlined above is dominated by the cost of the DFTs in Steps 5 and 7. We use the fast Fourier transform to implement the DFT. This means that we pad our data with zeros to force the dimension size of the grid to be a power of 2. At worst, our grid is $2rnx2rn$. The cost of the DFT is proportional to $4r^2n^2log2rn$. Even if r were as great as n, this cost would be proportional only to n^4logn. From an empirical standpoint, the algorithm outlined is faster for n (and k) of the order of 10.

The outlined algorithm places little limitation on the type of kernel function employed. Not only smooth, but also rough functions may comprise the basic objects from which the surface is built. We have used, inter alia, cones, hyperboloids, and Gaussian-shaped objects, some of which had fractal texture added to them. In Figures 1-4 we show profile plots. Figure 1 shows a real surface, Figure 2 the sampling grid we used to select data points. In Figure 2 the profiles depict what would have been obtained if we had used bilinear interpolation to build the surface. Figure 3 reveals the resultant surface when Gaussian kernel functions were used, while Figure 4 was obtained with a kernel function that had fractal texture added to a Gaussian base. When we compare the fitted surface to ground truth, the average error for the smooth kernel functions used by us, is approximately one percent of the data height range. As with any fitting technique, we cannot construct surface features that are not described by the sampled data.

We indicated that one reason for investigating global surface interpolation techniques was the need to calculate reliable estimates of surface curvature. In our preliminary trials with synthetic surface data the constructed surface appears to allow adequate surface curvature estimation. This will be tested further in future applications.

4. Conclusions

We have presented a method of surface interpolation that is computationally efficient. The reconstructed surface is fitted globally to enable the data rather than an implicit surface model to control the construction process. The method makes it possible to build not only the more customary smooth interpolated surface, but the roughly textured type as well.

Surface reconstruction methods provide a means of using the hypothesis-and-test approach in image analysis. They provide a mechanism for using image information that only constrains rather than specifies 3-D world parameters. The outlined algorithm is a tool for hypothesising a broad range of surface types.

References

1. Hardy, R.L., Multiquadric Equations of Topography and Other Irregular Surfaces, *J. Geophysical Res.*, Vol. 76, No. 8, 1971, pp. 1905-1915.
2. Stead, S.E., Smooth Multistage Multivariate Approximation, Ph.D. thesis, Division of Applied Mathematics., Brown University, Providence, Rhode Island, 1983.
3. Froberg, C-E., *Introduction to Numerical Analysis*, Addison-Wesley Publishing Co., 1969.
4. Hunt, B.R., The Application of Constrained Least Squares Estimation to Image Restoration by Digital Computer, *IEEE Trans. Computers*, Vol. C-22, No. 9, 1973, pp. 805-812.
5. Bracewell, R.N., *The Fourier Transform and Its Applications*, McGraw-Hill Book Co., 1978.

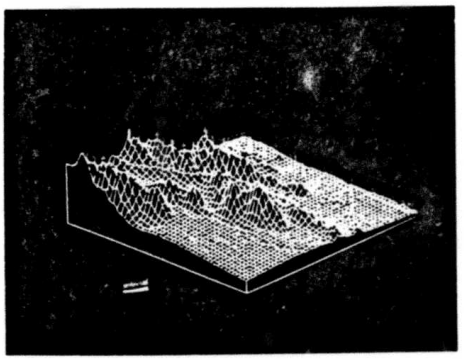


Figure 1. Real Surface Used as Ground Truth

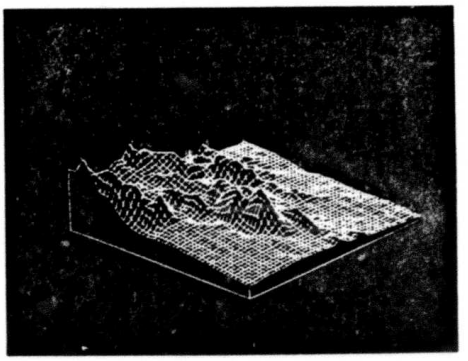


Figure 3. Reconstructed Surface by Means of Gaussian Kernel Functions.

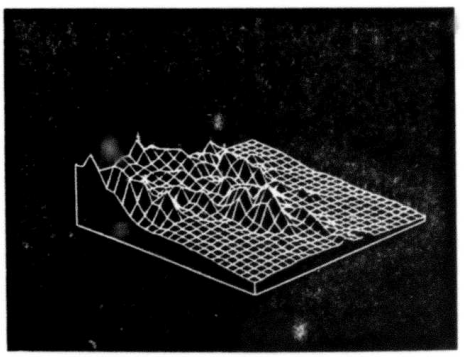


Figure 2. Data Sampled at Grid Intersections

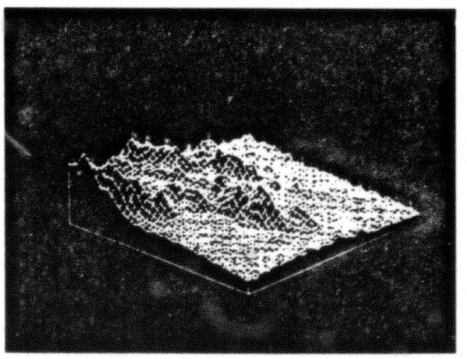


Figure 4. Reconstructed Surface by Means of Fractal Textured Gaussian Kernel Functions.

APPENDIX F

Hierarchical Warp Stereo
By: Lynn H. Quam

Hierarchical Warp Stereo

Lynn H. Quam
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

September 10, 1984

Abstract

This paper describes a new technique for use in the automatic production of digital terrain models from stereo pairs of aerial images. This technique employs a coarse-to-fine hierarchical control structure both for global constraint propagation and for efficiency. By the use of disparity estimates from coarser levels of the hierarchy, one of the images is geometrically warped to improve the performance of the cross-correlation-based matching operator. A newly developed surface interpolation algorithm is used to fill holes wherever the matching operator fails. Experimental results for the Phoenix Mountain Park data set are presented and compared with those obtained by ETL.

1 Introduction

The primary objective of this research was to explore new approaches to automated stereo compilation for producing digital terrain models from stereo pairs of aerial images. This paper presents an overview of the hierarchical warp stereo (HWS) approach , and shows experimental results when it is applied to the ETL Phoenix Mountain Park data set.

The stereo images are assumed to be typical aerial-mapping pairs, such as those used by USGS and DMA. Such pairs of images are different perspective views of a 3-D surface acquired at approximately the same time and illumination angles. Normally these views are taken with the camera looking straight downward. The major effect of non verticality is to increase the incidence of occlusion, which increases the difficulty of point correspondence.

We shall call one of these images the "reference image," and the other the "target image." We will be searching in the target image for the point that best matches a specified point in the reference image.

It is also assumed that the epipolar model for the stereo pair is known, which means that for any given point in one image we can determine a line segment in the other image that must contain the point, unless it is occluded from view by other points on the 3-D surface. This is certainly a reasonable assumption, since an approximation to the epipolar model can be derived from a relatively small number of point correspondences if the parameters of the imaging platform are not known a priori.

The primary goal is to automatically determine correspondences between points in the two images, subject to the following criteria:

- Minimize the rms difference between the disparity measurements and "ground truth." Without ground truth, we cannot measure this.
- Maximize the sensitivity of the disparity measurements to small-scale terrain features, while minimizing the effects of noise.
- Minimize the frequency of false matches.
- Minimize the frequency of match failures.

These criteria are mutually exclusive. Under ideal conditions, increasing the size of the match operator decreases the effects of noise on the disparity measurement, but it also diminishes sensitivity to small terrain features. Similarly, tightening the match acceptance criteria reduces the frequency of false matches, but results in more frequent match failures.

One of the goals of this system is to minimize the number of parameters that must be adjusted individually for each stereo pair to get optimum performance.

2 Approach

This section briefly explains the HWS approach, which consists of three major components:

- Coarse-to-fine hierarchical control structure for global constraint propagation as well as for efficiency.
- Disparity surface interpolation to fill holes wherever the matching operator fails.
- Geometric warping of the target image by using disparity estimates from coarser levels of the hierarchy to improve the performance of the cross-correlation-based matching operator.

2.1 The Use of Hierarchy and Surface Interpolation to Propagate Global Constraints

The goal of stereo correspondence is to find the point in the target image that corresponds to the same 3-D surface point as a given point in the reference image. It is often impossible to select the correct match point with only the image information that is local to the given point in the reference image in combination with the image nformation along the epipolar line segment in the target image. When the 3-D surface contains a replicated pattern, there is the likelihood of match point ambiguity. Let us consider, for example, a stereo pair that contains a parking lot

This research was supported by the Defense Advanced Research Projects Agency under Contract No. MDA 903-83-C-0027.

with repetitive markings delimiting the parking spaces. Around the edges of the lot there are image points that can be matched unambiguously. Within the parking lot, ambiguity is likely, depending on the orientation of the repetititive patterns with the epipolar line. A successful stereo correspondence system must be able to use global match information to resolve local match-point ambiguity.

HWS approaches this problem in two ways. First, global constraints on matches are propagated by the coarse-to-fine progression of the matching process. Disparities computed at lower resolution are employed to constrain the search in the target image to a small region of the epipolar line, which also greatly reduces the probability of selecting the wrong point when ambiguity is present. Second, whenever the match process fails to find a suitable match or detects a possible match ambiguity, a disparity estimate is inserted that is based on a surface interpolation algorithm, which uses information from a neighborhood around the disparity "hole," with the size of the neighborhood depending on the number of neighboring "holes."

2.2 The Use of Image Warping to Improve Correlation Operator Performance

One of the greatest problems in the use of area correlation for match point determination is the distortion that occurs because of disparity changes within the correlation window. Since area-based correlation matches areas, rather than individual points, the disparity it calculates is influenced by the disparities of all of the points in the window, not just the point at the center. When there are high disparity gradients or disparity discontinuities, the correlation calculated for the correct disparity can actually be so poor that some other disparity will have a higher correlation score.

The effect of correlation window distortion can be greatly mitigated in a hierarchical system by using the disparity estimates *from the previous level of matching to warp the target* image geometrically at its current resolution level into closer correspondence with the reference image.

2.3 Related Work

Norvelle [1] implemented a semi automatic stereo compilation system at the U.S. Army Engineer Topographic Laboratories (ETL) that operates in a single pass through the images. It uses disparity surface extrapolation both to predict the region of the epipolar segment for matching and to estimate the local surface orientation so as to warp the correlation window. He found that these techniques improved the performance of the system significantly, but that considerable manual intervention was needed when the surface extrapolator made bad predictions, or when the image contained areas with no information for matching, with ambiguities, or with occlusions.

3 Sequence Of Operations In Hierarchical Warp Stereo

Figure 1 illustrates the hierarchical control structure of the system.

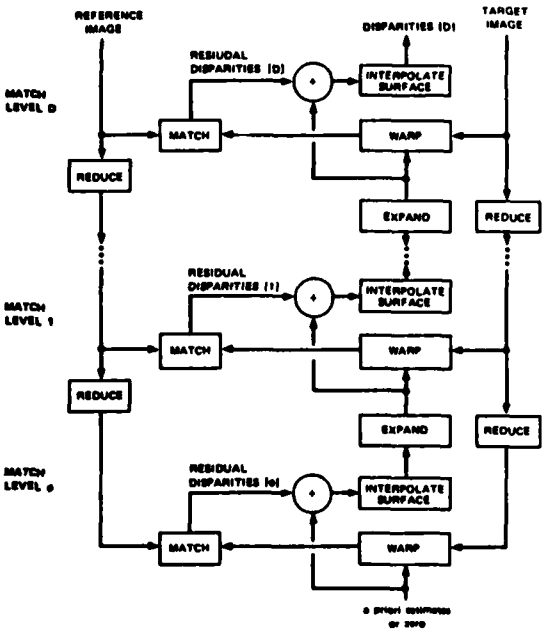


FIGURE 1

Block Diagram of Hierarchical Control Structure

1. Initialise:
   - Start with a stereo pair of images (assumed to be of the same dimensions).
   - Call one of these images the "reference image," the other the "target image."
   - Construct Gaussian pyramids (Burt [2]) $reference_i$ and $target_i$ for each image. The images at level i in these pyramids correspond to reductions of the original images by a factor of 2^i.
   - Set $disp_{-1}$ to either the a priori disparity estimates or all zeros.
   - Start the iteration at level $i = 0$.
   - Choose the pyramid depth D so that:

     $$D = ceiling(log2(uncertainty)) - 1.$$

     where *uncertainty* is an estimate of the maximum difference between $disp_{-1}$ and the "true" disparities. This guarantees the "true" disparities will be within the range (-2 : +2) at level 0 of the matching.
2. Warp: Use the disparity estimates $2 * disp_{i-1}$ to warp $target_{D-i}$ geometrically into approximate alignment with $reference_{D-i}$. Note that the factor of two is equal to the ratio of image scales between level i and level $i-1$ of the hierarchy.
3. Match: Using the matching operator, compute the residual disparities $\Delta disp_i$ between the warped target and the reference images at level i.

4. Refine: Compute the refined disparity estimates:

$$disp_i = 2 * disp_{i-1} + \Delta disp_i.$$

5. Fill: Use the surface interpolation algorithm to fill in disparities estimates at positions where matching operator fails because of no image contrast, ambiguity, etc.

6. Increase resolution: If $i = D$, quit; otherwise let $i = i + 1$ and go to Step 2.

4 Disparity Estimation

Disparity estimation consists of three parts:

- Computing match operator scores for disparities along an epipolar segment.
- Accepting or rejecting the collection of scores according to a model for the shape of the correlation peak.
- Estimating the subpixel disparities at acceptable peaks.

4.1 Match Score Operator

The HWS approach presented here can be implemented with a variety of match operators. All results reported here were obtained with an operator that closely approximates Gaussian-weighted normalized cross correlation. The values of the Gaussian weights decrease with Euclidean distance from the center of a square correlation window. In the examples shown here, the window dimension is 13×13 pixels with a standard deviation of approximately 2 pixels in the Gaussian weights. Preliminary results indicate that the Gaussian-weighted correlation operator is better than uniformly weighted correlation operators at locating changes in disparity while maintaining a given level of disparity precision.

4.2 Evaluation of Correlation Surface Shape

The match operator reports a failure if any of the following conditions exist:

- Disparity out of range: The maximum match score is found at either extreme of the epi-polar segment.
- Multiple peaks: The best and next best match scores is found at disparities that differ by more than one pixel.

There are other models for the expected shape of the correlation surface that can be based on the autocorrelation surface shape of the windows in the reference and target images. Further investigation is needed to evaluate the utility of such models for both surface shape evaluation and disparity estimation.

4.3 Subpixel Disparity Estimation

The subpixel location of the correlation surface peak is estimated by parabolic interpolation of both the x and y directions of disparity. For each direction, three adjacent match scores – s_{i-1}, s_i, and s_{i+1}, where s_i is the maximum score – are used to compute the peak as follows:

$$.5 * \frac{s_{i+1} - s_{i-1}}{2 * s_i - s_{i+1} - s_{i-1}}$$

More complicated approaches to peak estimation, such as two-dimensional least-squares fitting of the correlation surface, might yield better estimates, but at a higher computational cost.

5 Surface Interpolation Algorithm

The goal of the surface interpolation algorithm is to estimate values for the disparity surface at points where the match operator reported failure; such points will be called "holes." The approach to filling a hole at location x, y is to model the surface by employing the disparity measurements over the set of nonholes $\overline{H}$ in the $n \times n$ pixel neighborhood centered at x, y. The set H contains the indices of all holes in the neighborhood.

This surface interpolation algorithm is based on the solution to the hyperbolic multiquadric equations described in Smith [3]. The surface is known at the set of points x_i, y_i, z_i where $i \in \overline{H}$, and can be estimated at other points $h \in H$ by the formula

$$z(x_h, y_h) = \sum_{i \in \overline{H}} c_i * g(x_h - x_i, y_h - y_i),$$

where g is the basis function for the surface respresentation, and coefficients c_i are the solutions to the set of linear equations:

$$z(x_j, y_j) = \sum_{i \in \overline{H}} c_i * g(x_i - x_j, y_i - y_j) \quad for\ all\ j \in \overline{H}$$

Clearly, this irregular grid solution could be used to compute the surface values at the holes in the disparity data, but this involves solving for the coefficients c_j for each different configuration of holes and nonholes in the $n \times n$ neighborhoods of the disparity surface.

An alternative approach, which is used here, is to convert the quasi-regular grid problem into a regular grid problem in which each c_i at a hole is forced to be zero, and the corresponding z_i remains as an unknown. This results in the same solution that would have been obtained from the irregular grid formulation and produces the following system of linear equations:

$$\sum_{i \in H} A^{-1}_{h,i} * z_i = -\sum_{j \in \overline{H}} A^{-1}_{h,j} * z_j \quad for\ all\ h \in H, \qquad (1)$$

where A^{-1} is the inverse of the matrix $A_{i,j} = g(x_i - x_j, y_i - y_j)$ for $i, j \in H \cup \overline{H}$. This system of equations must be solved for each z_i for $i \in H$. Thus, we have reduced the size of the linear system of equations that must be solved from the number of elements in $\overline{H}$ to the number of elements in H. Of course, the matrix A must be computed and inverted once.

Areas on the disparity surface that contain large clusters of holes cause problems. The previous surface interpolation algorithm degenerates to a surface extrapolation algorithm when the nonholes in the neighborhood are not more or less isotropically distributed over the entire neighborhood. The problem can be overcome by increasing the size of the neighborhood until some spatial-distribution criterion is met, but this would require solving extremely large linear systems.

Large holes are filled by means of the following hierarchical approach:

Procedure Surface-Interpolate($surface_i$)

1. If $surface_i$ contains large holes then

(a) Compute filled-surface$_{i+1}$ = *expand*(surface-interpolate(*reduce*($surface_i$))), where *reduce* computes a Gaussian convolution reduction by a factor of two, surface-interpolate is a recursion call to this interpolation algorithm, and *expand* computes expansion by a factor of two, using bilinear interpolation.

(b) For each hole in $surface_i$ that is completely surrounded by other holes, fill the hole with the value from the filled-surface$_{i+1}$.

2. For each hole in $surface_i$ fill the hole by solving the system of linear equations (1) for the $n \times n$ pixel neighborhood centered at the hole (n = 7 in the examples).

3. Return the filled $surface_i$.

6 Examples

This section describes the experimental results achieved when the HWS technique was applied to areas of the ETL Phoenix Mountain Park data set, and compares these results to those obtained from the semiautomatic system developed by Norvelle [1].

The following components of the Phoenix Mountain Park data set were used:

- Left image: 2048 x 2048 pixels, 8 bits per pixel
- Right image: 2048 x 2048 pixels, 8 bits per pixel
- x-correspondence array: 400 x 400 points , floating point.

The left and right images had been scanned such that the epipolar lines were almostly exactly horizontal. The ETL x-correspondence array was converted to an x-disparity image to enable comparison between ETL and HWS results.

Results are shown for two different areas of the Phoenix data set. All disparity measurements are indicated in terms of pixel distances in the 2048 x 2048 Phoenix stereo pair, rather than the resolution of the selected windows.

- Area A is defined by two approximately aligned 150 x 150-pixel windows of the Phoenix pairs which were reduced by a factor of four (the windows thus corresponding to the 600 x 600-pixel windows of the originals). The measured disparities for area A range from -40 to +16 pixels.
- Area B is defined by two approximately aligned 125 x 125-pixel windows of the Phoenix pairs which were reduced by a factor of two (the windows thus corresponding to the 250 x 250-pixel windows of the originals). The measured disparities for area B range from -40 to -34 pixels.

Figures 2 and 3 show the inputs and outputs of three levels of the hierarchy for areas A and B, respectively. Columns 1 and 2 are the reference and target images at each level. Column 3 is a binary image that indicates the positions of match failures. Column 4 shows the resulting disparity image of each level after the match failures have been replaced by surface-interpolated disparity values.

Figures 4 and 5 contain a comparison of the HWS results with those obtained at ETL by Norvelle for areas A and B respectively. The bottom-left images of figures 4 and 5 show the pixel-by-pixel differences, after contrast enhancement, between the HWS and ETL disparities. The graphs to the right of these difference images depict the histograms of these differences.

The mean and standard-deviation values shown with the histograms provide a useful quantative comparision between the HWS and ETL results. They show that the average disparity differences were .082 and .025 pixels, and that the standard deviations of the disparity differences were .67 and .34 pixels for the A and B window pairs, respectively, in terms of pixel distances in the 2048 x 2048 Phoenix pairs. These standard deviations become .17 and .17 pixels when expressed relative to the scales of A and B windows, respectively.

Similar results have been achieved for other examples that include both higher resolution and larger windows.

7 Problems

HWS is still very experimental. Some of the parameters that affect the system, such as the range of disparities to compute at each level of hierarchy and the size of the correlation operator, are still specified manually.

There are problems in estimating the range of disparities to be computed at each level of the hierarchy. If the estimate is too low, there will be frequent out-of-range match failures. If, on the other hand, the estimate is too high, computation time will increase and there will be more potential for match point ambiguity.

HWS has difficulty dealing with steep terrain features that have small image projections, but large disparities. At low resolutions in the matching hierarchy, the disparities of the terrain surrounding the feature dominate those of the feature itself, resulting in a disparity estimate that is usually intermediate between that of the feature and that of the surround. At higher resolutions in matching, the disparity of the steep feature may be outside the permissible disparity range.

HWS has even greater problems with oblique stereo pairs containing many occlusions. At low matching resolution, the disparities of foreground and background in the same neighborhoods cannot be distinguished. As the matching resolution increases, foreground and background features are discernible as separate objects, but their disparities are out of range for the matcher.

Most of the difficulties caused by sudden changes in disparity might be solved by preceding the disparity surface interpolation step with an algorithm that attempts to match still unmatched regions in the reference image with regions in the target image that likewise have not yet been matched. We thus attempt to match holes with holes.

8 Conclusions

HWS produces very good results for vertical stereo pairs of rolling terrain. With the incluson of a hole-to-hole matching step, HWS should be capable of comparable performance for terrain characterized by steep slopes and frequent occlusions.

Bibliography

[1] Norvelle, F. Raye, *Interactive Digital Correlation Techniques For Automatic Compilation of Elevation Data*, U.S. Army Engineer Topographic Laboratories, Fort Belvoir, VA 22060, Report Number ETL-0272, Oct. 1981.

[2] Burt, Peter J., *Fast Filter Transforms for Image Processing*, CGIP, 16, 20-51. 1981.

[3] Smith, Grahame .B., *A Fast Surface Interpolation Technique*, Technical Note 333, Artificial Intelligence Center, SRI International, Menlo Park, California, August 1984. (Also in these proceedings).

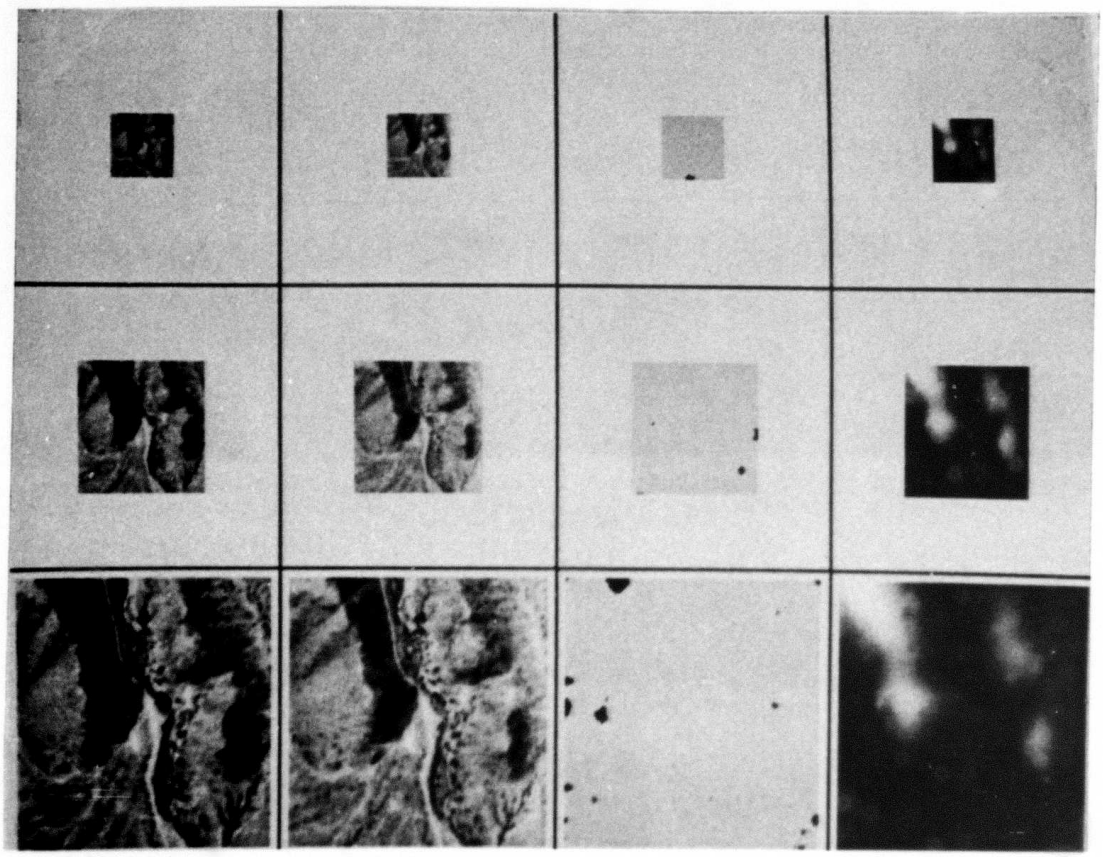
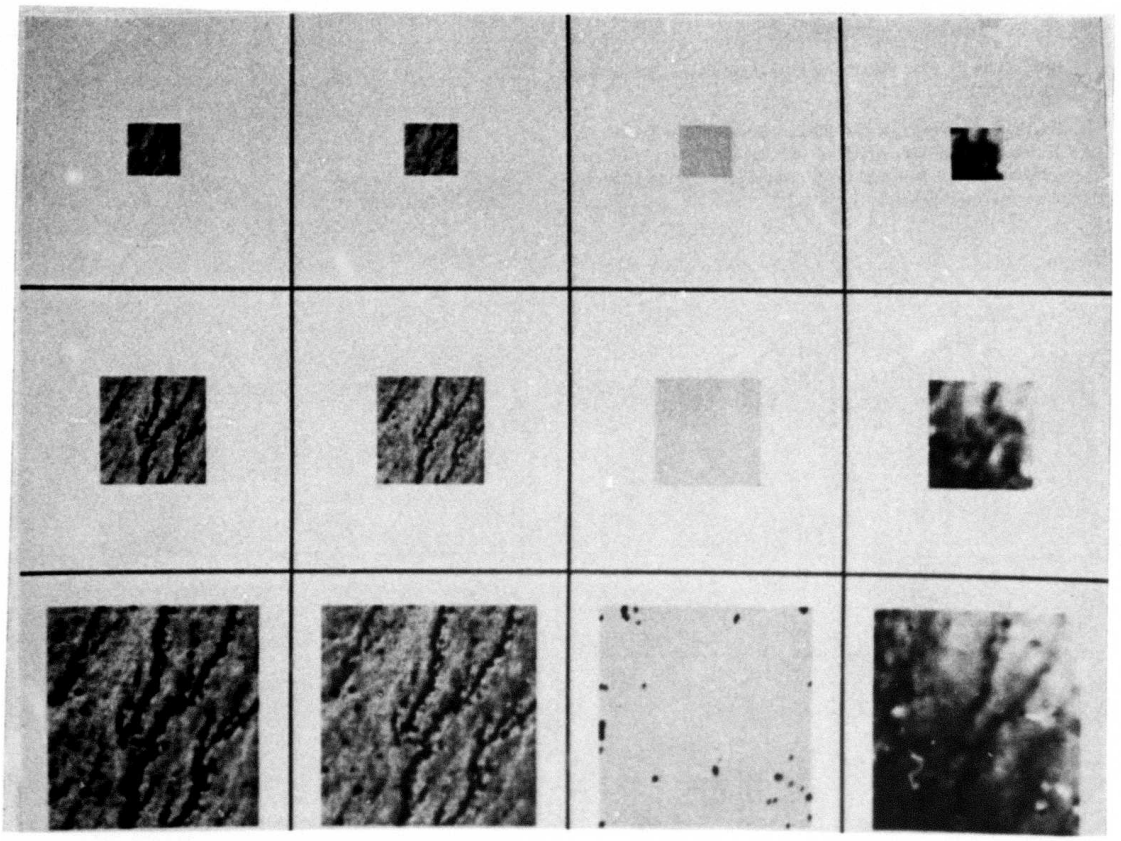
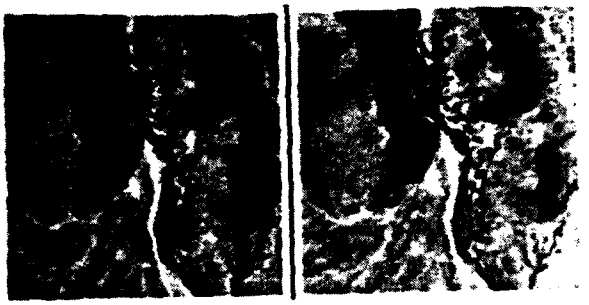


FIGURE 2 HWS results for area A

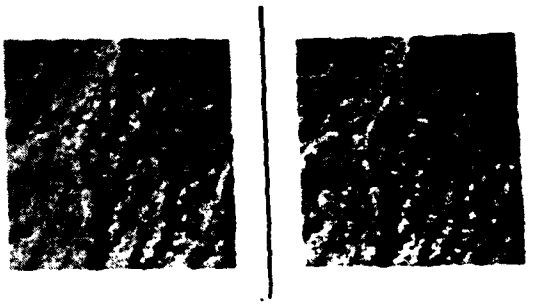


Reference images Target images Hole images Filled disparity images

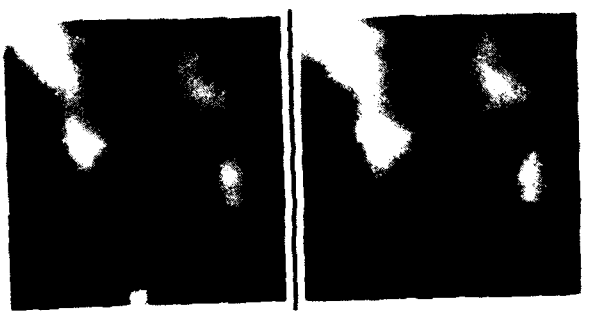
FIGURE 3 HWS results for area B



Reference image Target image

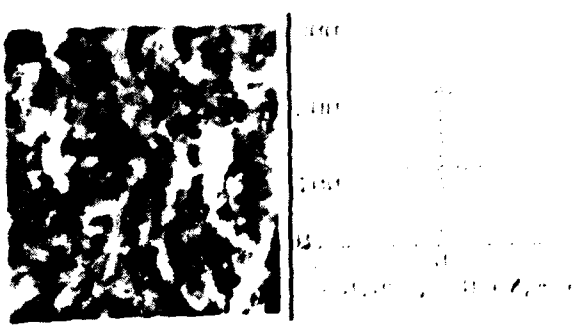


Reference image Target image



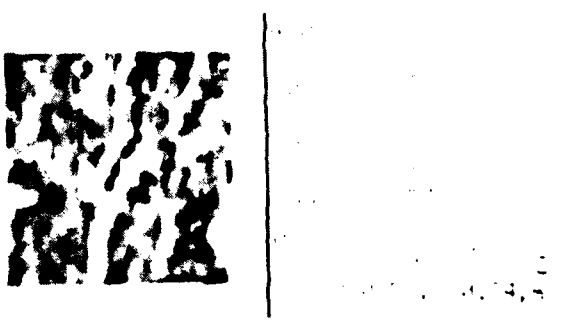
HWS disparity image ETL disparity image

HWS disparity image ETL disparity image



HWS - ETL difference HWS - ETL histogram

FIGURE 4 HWS vs. ETL results for area A



HWS - ETL difference HWS - ETL histogram

FIGURE 5 HWS vs. ETL results for area B

APPENDIX G

Shading into Texture
By: Alex P. Pentland

SHADING INTO TEXTURE

Alex P. Pentland
Artificial Intelligence Center, SRI International
333 Ravenswood Ave., Menlo Park, California 94025

ABSTRACT

Shape-from-shading and shape-from-texture methods have the serious drawback that they are applicable only to smooth surfaces, while *real surfaces* are often rough and crumpled. To extend such methods to real surfaces we must have a model that also applies to rough surfaces. The fractal surface model [Pentland 83] provides a formalism that is competent to describe such natural 3-D surfaces and, in addition, is able to predict human perceptual judgments of smoothness versus roughness — thus allowing the reliable application of shape estimation techniques that assume smoothness. This model of surface shape has been used to derive a technique for 3-D shape estimation that treats shading and texture in a unified manner.

I. INTRODUCTION

The world that surrounds us, except for man-made environments, is typically formed of complex, rough, and jumbled surfaces. Current representational schemes, in contrast, employ smooth, analytical primitives — e.g., generalized cylinders or splines — to describe three-dimensional shapes. While such smooth-surfaced representations function well in man-made, carpentered environments, they break down when we attempt to describe the crenulated, crumpled surfaces typical of natural objects. This problem is most acute when we attempt to develop techniques for recovering 3-D shape, for how can we expect to extract 3-D information in a world populated by rough, crumpled surfaces when all of our models refer to smooth surfaces only? The lack of a 3-D model for such naturally occurring surfaces has generally restricted image-understanding efforts to a world populated exclusively by smooth objects, a sort of "Play-Doh" world [1] that is not much more general than the blocks world.

Standard shape-from-shading [2,3] methods, for instance, all employ the heuristic of "smoothness" to relate neighboring points on a surface. Shape-from-texture [4,5] methods make similar assumptions: their models are concerned either with markings on a smooth surface, or discard three-dimensional notions entirely and deal only with ad hoc measurements of the image. Before we can reliably employ such techniques in the *natural* world, we must be able to determine which surfaces are smooth and which are not — or else generalize our techniques to include the rough, crumpled surfaces typically found in nature.

To accomplish this, we must have recourse to a 3-D model competent to describe both crumpled surfaces and smooth ones. Ideally, we would like a model that captures the intuition that smooth surfaces are the limiting case of rough, textured ones, for such a model might allow us to formulate a unified framework for obtaining shape from both shading (smooth surfaces) and texture (rough surfaces, markings on smooth surfaces).

* The research reported herein was supported by National Science Foundation Grant No. DCR-83-12766 and the Defense Advanced Research Projects Agency under Contract No. MDA 903-83-C-0027 (monitored by the U.S. Army Engineer Topographic Laboratory)

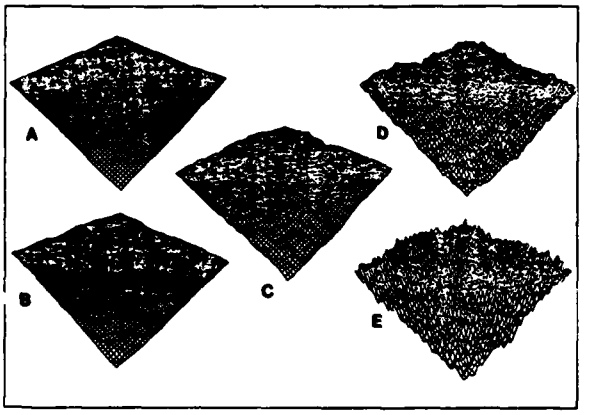


Figure 1. Surfaces of Increasing Fractal Dimension.

The fractal model of surface shape [6,7] appears to possess the required properties. Evidence for this comes from recently conducted surveys of natural imagery [6,8]. These survey found that the fractal model of imaged 3-D surfaces furnishes an accurate description of most textured and shaded image regions. Perhaps even more convincing, however, is the fact that fractals *look* like natural surfaces [9,10,11]. This is important information for workers in computer vision, because the natural appearance of fractals is strong evidence that they capture all of the perceptually relevant shape structure of natural surfaces.

II. FRACTALS AND THE FRACTAL MODEL

During the last twenty years, Benoit B. Mandelbrot has developed and popularized a relatively novel class of mathematical functions known as *fractals* [9,10]. Fractals are found extensively in nature [9,10,12]. Mandelbrot, for instance, shows that fractal surfaces are produced by many basic physical processes. The defining characteristic of a fractal is that it has a *fractional dimension*, from which we get the word "fractal." One general characterization of fractals is that they are the end result of physical processes that modify shape through local action. After innumerable repetitions, such processes will typically produce a fractal surface shape.

The fractal dimension of a surface corresponds quite closely to our intuitive notion of roughness. Thus, if we were to generate a series of scenes with the same 3-D relief but with increasing fractal dimension D, we would obtain a sequence of surfaces with linearly increasing perceptual roughness, as is shown in Figure 1: (a) shows a flat plane ($D \approx 2$), (b) rolling countryside ($D \approx 2.1$), (c) an old, worn mountain range ($D \approx 2.3$), (d) a young, rugged mountain range ($D \approx 2.5$), and, finally (e), a stalagmite-covered plane ($D \approx 2.8$).

EXPERIMENTAL NOTE: *Ten naive subjects (natural-*

language researchers) were shown sets of fifteen 1-D curves and 2-D surfaces with varying fractal dimension but constant range (e.g., see Figure 1), and asked to estimate roughness on a scale of one (smoothest) to ten (roughest). The mean of the subject's estimates of roughness had a nearly perfect 0.98 correlation (i.e., 96% of the variance was accounted for) ($p < 0.001$) with the curve's or surfaces's fractal dimension. The fractal measure of perceptual roughness is therefore almost twice as accurate as any other reported to date, e.g., [13].

Fractal Brownian Functions. Virtually all fractals encountered in physical models have two additional properties: (1) each segment is statistically similar to all others; (2) they are statistically invariant over wide transformations of scale. The path of a particle exhibiting Brownian motion is the canonical example of this type of fractal; the discussion that follows, therefore, will be devoted exclusively to fractal Brownian functions, which are a mathematical generalization of Brownian motion.

A random function $I(x)$ is a fractal Brownian function if for all x and Δx

$$Pr\left(\frac{I(x+\Delta x)-I(x)}{\|\Delta x\|^H} < y\right) = F(y) \qquad (1)$$

where $F(y)$ is a cumulative distribution function [7]. Note that x and $I(x)$ can be interpreted as vector quantities, thus providing an extension to two or more topological dimensions. If $I(x)$ is scalar, the fractal dimension D of the graph described by $I(x)$ is $D = 2-H$. If $H = 1/2$ and $F(y)$ comes from a zero-mean Gaussian with unit variance, then $I(x)$ is the classical Brownian function.

The fractal dimension of these functions can be measured either directly from $I(x)$ by using* of Equation 1, or from $I(x)$'s Fourier power spectrum** $P(f)$, as the spectral density of a fractal Brownian function is proportional† to f^{-2H-1}.

Properties of Fractal Brownian Functions. Fractal functions must be stable over common transformations if they are to be useful as a descriptive tool. Previous reports [6,7] have shown that the fractal dimension of a surface is invariant with respect to linear transformations of the data and to transformations of scale. Estimates of fractal dimension, therefore, may be expected to remain stable over smooth, monotonic transformations of the image data and over changes of scale.

A. The Fractal Surface Model And The Imaging Process

Before we can use a fractal model of natural surfaces to help us understand images, we must determine how the imaging process maps a fractal surface shape into an image intensity surface. The first step is to define our terms carefully.

DEFINITION: A **fractal Brownian surface** is a continuous function that obeys the statistical description given by Equation (1), with x as a two-dimensional vector at all scales (i.e., values of Δx) between some smallest (Δx_{min}) and largest (Δx_{max}) scales.

DEFINITION: A **spatially isotropic fractal Brownian surface** is a surface in which the components of the surface normal $\mathbf{N} = (N_x, N_y, N_z)$ are themselves fractal Brownian surfaces of identical fractal dimension.

Our previous papers [6,7] have presented evidence showing that most natural surfaces are spatially isotropic fractals, with Δx_{min} and Δx_{max} being the size of the projected pixel and the size of the examined surface patch, respectively. This finding has since been confirmed by others [8]. Furthermore, it is interesting to note that practical fractal generation techniques, such as those used in computer graphics, have had to constrain the fractal-generating function to produce spatially isotropic fractal Brownian surfaces in order to obtain realistic imagery [11]. Thus, it appears that many real 3-D surfaces are spatially isotropic fractals, at least over a wide range of scales* .

With these definitions in hand, we can now address the problem of how 3-D fractal surfaces appear in the 2-D image.

Proposition 1. A 3-D surface with a spatially isotropic fractal Brownian shape produces an image whose intensity surface is fractal Brownian and whose fractal dimension is identical to that of the components of the surface normal, given a Lambertian surface reflectance function and constant illumination and albedo.

This proposition (proved in [7]) demonstrates that the fractal dimension of the surface normal dictates the fractal dimension of the image intensity surface and, of course, the dimension of the physical surface. Simulation of the imaging process with a variety of imaging geometries and reflectance functions indicates that this proposition will hold quite generally; the "roughness" of the surface seems to dictate the "roughness" of the image. If we know that the surface is homogeneous,** we can estimate the fractal dimension of the surface by measuring the fractal dimension of the image data. What we have developed, then, is a method for inferring a basic property of the 3-D surface — i.e., its fractal dimension — from the image data. The fact that fractal dimension has also been shown to correspond closely to our intuitive notion of roughness confirms the fundamental importance of the measurement.

EXPERIMENTAL NOTE: *Fifteen naive subjects (mostly language researchers) were shown digitized images of eight natural textured surfaces drawn from Brodatz [14]. They were asked "if you were to draw your finger horizontally along the surface pictured here, how rough or smooth would the surface feel?" — i.e., they were asked to estimate the 3-D roughness/smoothness of the viewed surfaces. A scale of one (smoothest) to ten (roughest) was used to indicate 3-D roughness/smoothness. The mean of the subject's estimates of 3-D roughness had an excellent 0.91 correlation (i.e., 83% of the variance accounted was for) ($p < 0.001$) with roughnesses predicted by use of the image's 2-D fractal dimension and Proposition 1. This result supports the general validity of Proposition 1.*

B. Identification of Shading Versus Texture

Fractal functions with $H \approx 0$ are planar except for random variations described by the function $F(y)$ in Equation (1). If the variance of $F(y)$ is small people judge these surfaces to be "smooth"; thus, the fractal model with small values of H is appropriate for modeling smooth, shaded regions of the image. If the surface has significant local

*We rewrite Equation (1) to obtain the following description of the manner in which the second-order statistics of the image change with scale: $E(|\Delta I_{\Delta x}|)/\|\Delta x\|^{-H} = E(|\Delta I_{\Delta x=1}|)$ where $E(|\Delta I_{\Delta x}|)$ is the expected value of the change in intensity over distance Δx. To estimate H, and thus D, we calculate the quantities $E(|\Delta I_{\Delta x}|)$ for various Δx, and use a least-squares regression on the log of our rewritten Equation (1).

**That is, since the power spectrum $P(f)$ is proportional to f^{-2H-1}, we may use a linear regression on the log of the observed power spectrum as a function of f (e.g., a regression using $\log(P(f)) = -(2H+1)\log(f)+k$ for various values of f) to determine the power H and thus the fractal dimension.

†Discussion of the rather technical proof of this proportionality may be found in Mandelbrot [10].

*This does not mean that the surfaces are completely isotropic, mearly that their fractal (metric) properties are isotropic.

**Perhaps determined by the use of imaged color.

fluctuations, i.e., if $F(y)$ is large, the surface is seen as being smooth but textured, in the sense that markings or some other 2-D effect is modifing the appearance of the underlying smooth surface. In contrast, fractals with $H > 0$ are not perceived as smooth, but rather as being rough or three-dimensionally textured.

The fractal model can therefore encompass shading, 2-D texture, and 3-D texture, with shading as a limiting case in the spectrum of 3-D texture granularity. The fractal model thus allows us to make a reasonable, rigorous and perceptually plausible definition of the categories "textured" *versus* "shaded," "rough" *versus* "smooth," in terms that can be measured by using the image data.

The ability to differentiate between "smooth" and "rough" surfaces is critical to the performance of current shape-from-shading and shape-from-texture techniques. For surfaces that, from a perceptual standpoint, are smooth ($H \approx 0$) and not 2-D textured ($Var(F(y))$ small), it seems appropriate to apply shading techniques.* For surfaces that have 2-D texture it is more appropriate to apply available texture measures. Thus, use of the fractal surface model to infer qualitative 3-D shape (namely, smoothness/roughness), has the potential of significantly improving the utility of many other machine vision methods.

III. Shape Estimates From Texture And Shading

The fractal surface model allows us to do quite a bit better than simply identifying smooth *versus* textured surfaces and applying previously discovered techniques. Because we have a unified model of shading, 2-D texture and 3-D texture, we can derive a shape estimation procedure that treats shaded, two-dimensionally textured, and three-dimensionally textured surfaces in a single, unified manner.

A. Development of a Robust Texture Measure

Let us assume that: (1) albedo and illumination are constant in the neighborhood being examined, and (2) the surface reflects light isotropically (Lambert's law). We are then led to this simple model of image formation:

$$I = \rho\lambda(\mathbf{N} \cdot \mathbf{L}) \tag{2}$$

where ρ is surface albedo, λ is incident flux, $\mathbf{N}$ is the [three-dimensional] unit surface normal, and $\mathbf{L}$ is a [three-dimensional] unit vector pointing toward the illuminant. The first assumption means that the model holds only within homogeneous regions of the image, e.g., regions without self-shadowing. The second assumption is an idealization of matte, diffusely reflecting surfaces and of shiny surfaces in regions that are distant from highlights and specularities [3].

In Equation (2), image intensity is dependent upon the surface normal, as all other variables have been assumed constant. Similarly, the second derivative of image intensity is dependent upon the second derivative of the surface normal, i.e.,

$$d^2I = \rho\lambda(d^2\mathbf{N} \cdot \mathbf{L}) \tag{3}$$

(**Notation**: we will write d^2I and $d^2\mathbf{N}$ to indicate the second derivative quantities computed along some image direction (dx, dy) — this direction to be indicated implicitly by the context.)

The fractal model taken together with previous results [15], implies that on average $d^2\mathbf{N}$ is parallel to $\mathbf{N}$. Consequently, if we divide Equation (2) by Equation (3) we will *on average* obtain the following relationship:

$$E\left(\left|\frac{d^2I}{I}\right|\right) = E\left(\left|\frac{\rho\lambda(d^2\mathbf{N} \cdot \mathbf{L})}{\rho\lambda(\mathbf{N} \cdot \mathbf{L})}\right|\right) \approx E(\|d^2\mathbf{N}\|) \tag{4}$$

where $E(x)$ denotes the expected value [mean] of x. That is, we can estimate how crumpled and textured the surface is (i.e., the average magnitude of the surface normal's second derivative) by observing $E(|d^2I/I|)$.

Equation (4) provides us with a measure of 3-D texture that is (on average and under the above assumptions) independent of illuminant effects. This measure is affected by foreshortening, however, which acts to increase the apparent frequency of variations in the surface, e.g., the average magnitude of $d^2\mathbf{N}$. We can, therefore, obtain an estimate of surface orientation by employing the approach adopted in other texture work [5]: if we assume that the 3-D surface texture is isotropic, the surface tilt* is simply the direction of maximum $E(|d^2I/I|)$ and the surface slant** can be derived from the ratio between $\max_\theta E(|d^2I/I|)$ and $\min_\theta E(|d^2I/I|)$, where θ designates the [implicit] direction along which the texture measure is evaluated. Specifically, the surface slant is the arc cosine of z_N, the z-component of the surface normal, and for isotropic textures z_N is equal to the square root of this ratio. The square-root factor is necessitated by the use of second-derivative terms.

One of the advantages of this shape-from-texture technique is that not only can it be applied to the 2-D textures addressed by other researchers [4,5] (by simply using this texture frequency measure in place of theirs†), but it can also be applied to surfaces that are three-dimensionally textured — and in exactly the same manner. This texture measure, therefore, allows us to extend existing shape-from-texture methods beyond 2-D textures to encompass 3-D textures as well.

B. Development of a Robust Shape Estimator

These shape-from-texture techniques are critically dependent upon the assumption of isotropy: when the textures are anisotopic (stretched), the error is substantial. Estimates of the fractal dimension of the viewed surface [6,7], by virtue of their independence with respect to multiplicative transforms, offer a partial solution to this problem. Because foreshortening is a multiplicative effect, the computed fractal dimension is not affected by the orientation of the surface.†† Thus, if we measure the fractal dimension of an isotropically textured surface along the x and y directions, the measurements must be identical. If, however, we find that they are unequal, we then have *prima facie* evidence of anisotropy in the surface.

This method of identifying anisotropic textures is most effective when each point on the surface has the same direction and magnitude of anisotropy, for in these cases we can accurately discriminate changes in fractal dimension between the x and y directions. When the surface texture is variable, however, this indicator of anisotropy becomes less useful. Thus, local variation in the surface texture remains a major source of error in our estimation techniques; it is therefore important to develop a method of estimating surface orientation that is robust with respect to local variation in the surface texture.

---

*Indeed, it is only in these cases that measurement noise can be reduced (by averaging) to the levels required by shape-from-shading techniques without simultaneously destroying evidence of surface shape.

*The image-plane component of the surface normal, i.e., the direction the surface normal would face if projected onto the image plane.

**The depth component of the surface normal.

†This measure includes edge information, i.e., the frequency of Marr-Hildreth zero-crossings as we move in a given direction appears to be proportional to $E(|d^2I/I|)$ along that direction; consider that Marr-Hildreth zero-crossings are also zero-crossings of d^2I/I.

††At least not until self-occlusion effects have become dominant in the appearance of the surface.

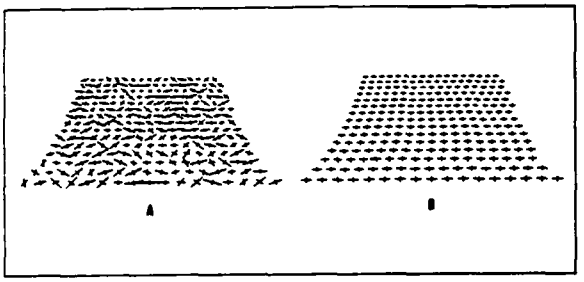


Figure 2. Variation in Local Texture (a) Compared with No Variation (b).

Such robustness can be obtained by applying regional, rather than purely local, constraints. Natural textures are often "homogeneous" over substantial regions of the image, although there may be significant local variation within the texture, because the processes that act to create a texture typically affect regions rather than points on a surface. This fact is the basis for interest in texture segmentation techniques. Current shape-from-texture techniques do not make use of the regional nature of textures, relying instead on point-by-point estimates. By capitalizing on the regional nature of textures we can derive a substantial additional constraint on our shape estimation procedure.

Let us assume that we are viewing a textured planar surface whose orientation is a 30° slant and a vertical tilt. Let us further suppose that the surface texture varies randomly from being isotropic to being anisotropic (stretched) up to an aspect ratio of 3:1, with the direction of this anisotropy also varying randomly. Such a surface, covered with small crosses, is shown in Figure 2(a); for comparison, the same surface, minus anisotropies, is shown in Figure 2(b).

If we apply standard shape estimation techniques — i.e., estimating the amount of foreshortening (and thus surface orientation) by the ratio of some texture measure along the [apparently] unforshortened and [apparently] maximally foreshortened directions — our estimates of the foreshortening magnitude will vary widely, with a mean error of $65^{\circ}_{\circ}$ and an rms error of $81^{\circ}_{\circ}$. If, however, we estimate the value α of the unforshortened texture measure by examining the entire region, and then compare this regional estimate to the texture measure along the (apparently) maximally foreshortened direction then our mean error is reduced to $40^{\circ}_{\circ}$ and the rms error to 49%.

By combining this notion of regional estimation with the texture measure developed above, i.e., $E(|d^2I/I|)$, we can construct the following shape-from-texture algorithm that is able to deal with both smooth two-dimensionally textured surfaces and rough, three-dimensionally textured surfaces, and that is robust with respect to local variations in the surface texture.

C. A Shape Estimation Algorithm

We may construct a rather elegant and efficient shape estimation algorithm based on the notion of regional estimation and on the texture measure introduced above by employing the fact that

$$\nabla^2 I = \frac{d^2 I}{du^2} + \frac{d^2 I}{dv^2} \tag{5}$$

for any orthogonal u, v. This identity will allow us to estimate the surface slant immediately rather than having to search all orientations for the directions along which we obtain the maximum and minimum values of $E(|d^2I/I|)$.

Let us assume that we have already determined $\alpha = \min_\theta E(|d^2I/I|)$, which is the regional estimate of unforeshortened $E(|d^2\mathrm{N}|)$. When the estimate of α is exact, Equation (5) gives us the result that

$$E\left(\left|\frac{\nabla^2 I}{I}\right|\right) - \alpha = \max_\theta E\left(\left|\frac{d^2 I}{I}\right|\right) \tag{6}$$

as the directions of maximum and minimum $E\left(\left|\frac{d^2I}{I}\right|\right)$ are orthogonal.

We may therefore estimate z_N, the z component of the surface normal, by

$$z_N = \left(\frac{\beta - \alpha}{\alpha}\right)^{-1/2} \tag{7}$$

where $\beta = E(|\nabla^2 I/I|)$ and α is the regional estimate of the unforeshortened value of $E(|d^2I/I|)$. The constant α can be estimated either by the median of the local [apparently] unforeshortened texture-measure values, or by use of the constraint that $0 \leq z_N \leq 1$ within the region. The direction of surface tilt can then be estimated by the gradient of the resulting slant field — e.g., the local gradient of the z_N values — or (as in other methods) by examining each image direction to find the one with the largest-value of the texture frequency measure. In actual practice we have found that the gradient method is more stable.

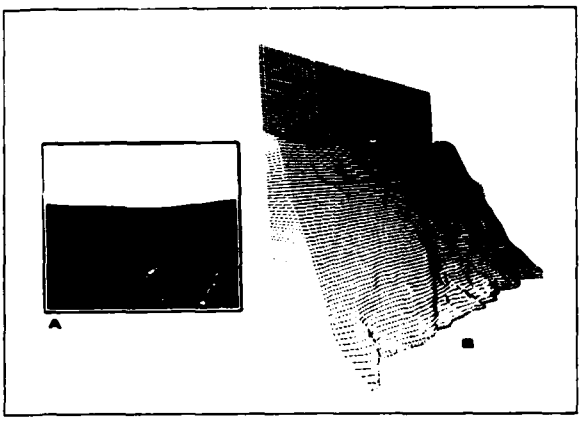


Figure 3. Tuckerman's Ravine.

D. A Unified Treatment of Shading and Texture

The fractal surface model captures the intuitive notion that, if we examine a series of surfaces with successively less three-dimensional texture, eventually the surfaces will appear shaded rather than textured. Because the shape-from-texture technique developed here was built on the fractal model, we might expect that it too would degrade gracefully into a shape-from-shading method. This is in fact the case: this shape-from-texture technique is identical to the local shape-from-shading technique previously developed by the author [15]. That is, we have developed a shape-from-x technique that applies equally to 2-D texture, 3-D texture and shading.

As an example of the application of this shape-from-texture-and-shading technique,* Figure 3 shows (a) the digitized image of Tuckerman's ravine (a skiing region on Mt. Washington in New Hampshire), and (b) a relief map giving a side view of the estimated surface shape, obtained by integrating the slant and tilt estimates.**

*This example was originally reported in Pentland [15] as the output of a local shape-from-shading technique followed by averaging and integration. This algorithm is identical to the shape-from-texture technique described here; in fact, investigation of the shape-from-texture properties of this method was motivated by the consternation caused by this successful application of a *shading* technique to a *textured* surface.

This relief map may be compared directly with a topographic map of the area; when we compare the estimated shape with the actual shape, we find that the roll-off at the top of Figure 3(b) and the steepness of the estimated surface are correct for this surface; the slope of this area of the ravine averages 60°.

IV. Summary

Shape-from-shading and texture methods have had the serious drawback that they are applicable only to smooth surfaces, while real surfaces are often rough and crumpled. We have extended these methods to real surfaces using the fractal surface model [6,7]. The fractal model's ability to distinguish successfully between perceptually "smooth" and perceptually "rough" surfaces allows reliable application of shape estimation techniques that assume smoothness. Furthermore, we have used the fractal surface model to construct a method of estimating 3-D shape that treats shading and texture in a unified manner.

REFERENCES

[1] H.G. Barrow and J.M. Tenenbaum, "Recovering Intrinsic Scene Characteristics From Images," in A. Hanson and E. Riseman, Eds., *Computer Vision Systems*, Academic Press, New York, New York 1978

[2] B. K. P. H. Horn, "Shape From Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View," A.I. Technical Report 79, Project MAC, M.I.T. (1970).

[3] B. K. P. H. Horn and K. Ikeuchi, "Numerical Shape from Shading and Occluding Boundaries," Artificial Intelligence, 15, Special Issue on Computer Vision, pp. 141-184 (1981).

[4] J. R. Kender, "Shape From Texture: An Aggregation Transform that Maps a Class of Textures Into Surface Orientation," Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (1979).

[5] A. P. Witkin, "Recovering Surface Shape and Orientation from Texture," *Artificial Intelligence, 17, pp. 17-47 (1981).*

[6] A. Pentland, "Fractal-Based Description," Proceedings of International Joint Conference on Artificial Intelligence (IJCAI) '83, Karlsruhe, Germany, August 1983.

[7] A. Pentland, "Fractal-Based Description Of Natural Scenes," IEEE Transactions on Pattern Analysis and Machine Intelligence, *to appear* September 1984.

[8] G. Medioni and Y. Yasumoto, "A Note on using the Fractal Dimension for Segmentation," IEEE Computer Vision Workshop, Annapolis, MD, April 30- May 3, 1984.

[9] B. B. Mandelbrot, "Fractals: Form, Chance and Dimension," W. H. Freeman and Co.,San Francisco, California, 1977.

[10] B. B. Mandelbrot, "The Fractal Geometry of Nature," W. H. Freeman, San Francisco, 1982.

[11] A. Fournier, D. Fussel and L. Carpenter, "Computer Rendering of Stochastic Models," Communications of the ACM, vol. 25, 6, pp. 371-384, 1982.

[12] L. F. Richardson, "The Problem of Contiguity: an Appendix of Statistics of Deadly Quarrels," General Systems Yearbook, vol. 6, pp. 139-187, 1961.

[13] H. Tamura, S. Mori, and T. Yamawaki, "Textural Features Corresponding to Visual Perception," IEEE Trans. on Sys., Man and Cyber., Vol. SMC-8, No. 6, pp.460-473, June 1978

[14] P. Brodatz, "Textures: A Photographic Album for Artists and Designers," Dover, New York, New York, 1966.

[15] Pentland, A. P. (1984), "Local Shape Analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, March 1984, pp. 170-187

**The shape algorithm produces estimates of the surface orientation. For display purposes, these estimates were integrated to produce a relief map of the surface.

APPENDIX H

Goal-Directed Textured-Image Segmentation
By: Kenneth I. Laws

Goal-Directed Textured-Image Segmentation

Kenneth I. Laws
Artificial Intelligence Center
SRI International

Abstract

The SLICE textured-image segmentation system identifies image regions that differ in gray-level distribution, color, spatial texture, or other local property. This report concentrates on textured-image segmentation using local texture-energy measures and user-delimited training regions. The SLICE algorithm combines knowledge of target textures with knowledge of background textures by using histogram-similarity transforms. Regions of high similarity to a target texture and of low similarity to any negative examples are identified and then mapped back to the original image. This use of texture-similarity transforms during the segmentation process improves segmenter performance and focuses segmentation activity on material types of greatest interest. The system can also be used for goal-independent texture segmentation by omitting the similarity-transform computations, and its hierarchical, recursive segmentation strategy integrates very well with other image-analysis techniques.

1. Introduction

This paper presents a new goal-directed method of textured-image segmentation. The SLICE *segmentation algorithm* is one component of a proposed knowledge-based image feature-extraction system. The algorithm is currently implemented in the SLICE *program*, a region-based recursive segmentation system running on the DARPA/DMA Image Understanding Testbed at SRI International. The SLICE program is capable of goal-independent segmentation and other image manipulations in addition to the texture segmentation discussed in this paper.

Aerial images are very difficult to segment into meaningful regions, despite the fact that humans seem to do this effortlessly. Attempts to develop segmentation algorithms using only monochrome input data have had little success. Segmentation using color and infrared

This research was supported by the Defense Advanced Research Projects Agency under Contract No. MDA903-83-C-0027.

data has worked somewhat better, but such data is often unavailable. This report describes techniques for using the spatial textures in monochrome imagery in much the same way that color has previously been used.

Image textures arise from many physical sources. Cellular textures are composed of repeated similar elements, such as leaves on a tree or bricks in a wall; other texture types include flow patterns, fiber masses, and stress cracks. A complete *analysis* of any texture would require modeling of the underlying physical structures and processes. In most applications, texture *recognition* is more important than knowledge of the generating mechanism. The algorithm presented in this report can be used for texture recognition and material identification when we have knowledge of scene texture types, and for locating and characterizing textured regions even when we have no such knowledge.

The SLICE algorithm consists of three parts: goal-directed texture transformation, multiple histogram-based threshold segmentations, and spatial analysis of the proposed segmentations in order to choose the best one. These steps may be repeated on the newly found regions to further segment them. A high-level control system could be used to focus the segmenter's "attention" on important image regions and can determine when to stop partitioning a given region (using size, shape, homogeneity, semantic, or priority considerations). Regions found by other image-analysis techniques can also be combined with those found by the SLICE algorithm.

This report describes the SLICE algorithm and the rationale for each part of the technique. Section 2 introduces some definitions used throughout the report. Section 3 briefly describes the basic texture transforms used to measure local spatial variation around a pixel. Section 4 discusses maximum likelihood classification methods, and points out why they are not optimal for texture segmentation. Section 5 presents similarity transforms that can be used to locate desired texture signatures in an image. Sections 6 and 7 describe the integration of texture similarity transforms with histogram-based segmentation to produce goal-directed segmentation using multiple texture bands. Section 8 then presents examples of the technique, and Section 9 summarizes the characteristics of this approach. Details of the modified PHOENIX goal-independent color-image segmentation technique used in the current SLICE program are presented in Appendices A and B.

2. Background

An image is a two-dimensional array of pixels, where pixels are numbers (usually integers in the range 0 to 255) or vectors of numbers representing information about an imaged scene. An image of vector-valued pixels may be thought of as a set of two-dimensional, scalar-valued layers called data bands. (Indeed, the pixel data is usually stored in this layered fashion.)

Pixel values typically represent intensity of light (infrared, visible, or ultraviolet) or other electromagnetic energy reaching a sensor from a point in the imaged scene, but may correspond to other measurable scene properties. Data bands may also record such computed information as stereo disparity, intensity gradients, filter responses, estimated scene albedo, or inferred surface slope. In this paper we shall be particularly concerned with texture bands computed from local texture properties around each pixel.

The integer values that can be assumed by a scalar pixel are called gray levels. Even non-

physical data values such as texture statistics are representable as gray levels or intensities, since the data bands may be displayed on an image monitor as black-and-white luminance images. A special type of data band, called a *map*, stores at each pixel an integer value representing the material type or other nominal category assigned to that pixel. A *segmentation map* or *region map* has a unique integer assigned to all pixels in an image region and different integers assigned to different regions. Segmentation maps are often displayed using pseudocolor (i.e., arbitrary assigned region colors), since display as a luminance image is not meaningful.

The value of a pixel is easily read out of the array; all other information about the imaged scene is implicit. It is the task of low-level image processing to make useful spectral or spatial information explicit so that the more expensive high-level feature-extraction operators and reasoning processes can utilize it. This paper will describe a goal-directed method for extracting homogeneous image regions satisfying prespecified criteria as to size, location, gray-level distribution, and texture.

3. Texture Transforms

Textures can be recognized if one or more distinctive properties can be measured. (There are also structural or "syntactic" pattern-recognition methods that do not require texture metrics.) Many ways of computing texture descriptors have been proposed. Some of the most powerful descriptors, both individually and in combination, are the *texture-energy* measures [Laws 80] and their variants [Pietikäinen 82, Harwood 83]. These measures do not describe texture-generating mechanisms or parameters directly, but do tend to be constant across any perceptually homogeneous texture region and distinct for distinct textures. (Within macrotextures having large elements they tend to be multimodal with histogram peaks corresponding to the edges and interiors of the texture elements.)

Texture energy is the amount of variation within a filtered window around a pixel. A particular texture energy measure thus depends on the spatial filter, the window size, and the method of measuring average variation within the window. The transforms require only simple convolution and moving-average techniques; moreover they can be made invariant to changes in image illumination, contrast, and rotation without histogram equalization or other preprocessing operations.

There are two required steps in applying a texture-energy transform. The first step is to filter the original scalar image with a small convolution mask. The mask is typically a binomially weighted array (defined below) that enhances image spots, edges, or high-frequency components. Binomially weighted masks are both separable and decomposable into smaller convolution masks, making them easy to implement efficiently on a variety of architectures. The set of filter masks used determines the spatial frequencies or texture structures that the transforms will measure.

The 3×3 binomially weighted masks are shown in Figure 1. They were constructed by convolving the vectors $[\ 1\ \ 2\ \ 1]$, $[-1\ \ 0\ \ 1]$, and $[-1\ \ 2\ -1]$ with their own transposes.[1] Larger masks may be constructed by convolving the 3×3 masks with themselves. Binomially

[1]These vectors are themselves constructed from the vectors $[\ 1\ \ 1]$ and $[-1\ \ 1]$. The mask names are derived from the terms *level*, *edge*, and *spot* for the 3-vectors of sequency 0, 1, and 2. Similar names are used for the 5-vectors and 5×5 masks, with the addition of W (*wave*) and R (*ripple*) for the vectors of sequency 3 and 4.

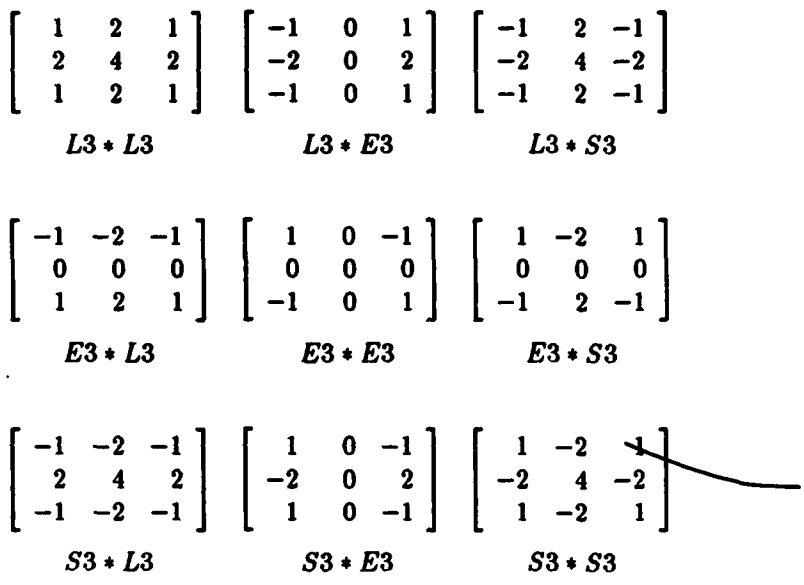


Figure 1: Orthogonal 3 × 3 Texture Masks

weighted filter masks of sizes 3 × 3, 5 × 5, and 7 × 7 were found to be nearly equivalent in tests on a very limited class of textures [Laws 80]. For aerial image analysis, the 3 × 3 masks seem likely to be the most useful, although strong patterns such as orchards and crop rows may be better discriminated using larger masks.

An exponential series of mask sizes may be needed for multiresolution texture analysis. Application of large binomial weighted masks (with coefficients in the billions) can be very difficult even if done by repeated filtering with smaller masks. A better method is to construct a pyramid of image reductions and then apply a single mask size to all levels of the pyramid. The unfiltered image itself may be used as the highest-resolution "filtered" band, and its local-energy statistics may be useful either as texture measures or for normalizing the other texture measures when contrast invariance is desired.

The second texture-transformation step is to apply a *local-energy* operator to the filtered image to produce a texture-energy data band. Texture energy at a point is just the variance of the filtered-image values computed over a window around the point. Standard deviation, or the square root of the variance, has been found just as effective. For zero-mean filtered bands, the standard deviation is usually approximated by an average of the filtered-image magnitudes (i.e., absolute values) over a window. Such averages can be computed by moving-window techniques that are very fast, even on general-purpose digital computers.

An energy-gathering window of about about five or ten times the area of the filter mask is recommended; larger sizes give better classification accuracy when applied to large texture patches but lack the resolution needed for analysis of typical 512 × 512 aerial image displays. The time required to compute the local energy is independent of the window size, since each pixel is examined only once as it enters the window and once as it leaves.

(A fading-memory approximation to moving-window averaging can also be used to permit single-pass computation of texture energy values without storing the intermediate filtered image.)

Texture descriptors computed with the suggested masks are unaffected by most scene illumination and sensor bias effects because all but the first mask produce zero-mean outputs. Level-invariant texture energy bands may be normalized by the variance of the unfiltered image if contrast-invariant texture measures are desired. Pairs of texture energy values representing directional image structure can also be averaged if rotation-invariant texture measures are desired. Decisions about when to normalize or average are best left to a control system capable of reasoning about particular analysis tasks and image contexts.

4. Maximum Likelihood: The Trouble with Optimal Discrimination

We now have a multiband image composed of luminance or spectral bands and derived texture bands. We want a quick way to partition the vector-valued pixels into homogeneous groups, preferably using a priori knowledge of target signatures when it is available. The texture transforms make it likely that each texture signature in the image will have a fairly predictable Gaussian distribution in at least one data band. The temptation to jump to multivariate Gaussian discriminant analysis is almost overwhelming.

There is a good reason for trying other methods, however, even when we have sufficient multivariate training data to compute the needed means and variances (or covariance matrices). Maximum-likelihood Gaussian discriminant analysis[2] is optimal for separating two or more multivariate Gaussian distributions, but we do not have Gaussian distributions as such—we have mixtures thereof. Even within a single data band we may have at best one Gaussian and one mixture density to be discriminated.

This is not to say that the discriminant analysis won't work, only that the conditions for optimality are not satisfied. The procedure for computing discriminant functions will reduce positive and negative training instances to means and standard deviations, reject any data bands in which the means and standard deviations are similar, and do the best it can with a linear or perhaps quadratic function of the means and standard deviations in the remaining bands. The result is that much of our knowledge about target signatures in different data bands will be discarded.

As an example, consider a texture that is known to have a Gaussian distribution in a particular data band. Assume that the scene might also contain instances of another texture with a strongly bimodal salt-and-pepper distribution. These two distributions should be easily distinguished, but they may not be discriminable by mean and standard deviation alone. We could use multivariate statistics and compute covariances with data values in other bands, but the resulting classifier might be unstable and difficult to train. We could also seek a transform to another data band in which the two textures are separable, but that will not work if there are other textures that might also be present in the scene. Discriminant analysis is thus a poor way to deal with this situation.

There is also the matter of a priori probability. Any form of maximum-likelihood classifier performs best if the decision thresholds are properly adjusted for the a priori probability of that texture's appearing in the scene. We may be able to guess reasonable probabilities

[2] I.e., multivariate minimum-distance classification using an inverse-covariance, or *Mahalanobis*, weighting.

based on previous analyses of similar imagery, but biasing the analysis with such values instead of just examining the evidence in the image is a suspect procedure. Assigning equal likelihood to all possible scene entities is even more suspect.

We could live with these problems, but there are better ways of finding and distinguishing textures. These will be described in the remainder of this paper.

5. Similarity Transforms: Encoding Goals and Knowledge

The key to efficient goal-directed segmentation is to estimate quickly whether any given pixel is part of the texture or target signature we are seeking. We have already computed texture bands in order to collapse implicit information about a pixel's neighborhood into explicit information stored in the pixel's data vector. We now need a scalar measure of similarity (or, conversely, of dissimilarity) between a pixel data vector and a target signature.

The simplest dissimilarity measure is the Euclidean distance between the image pixel vector and a prototype vector representing a known texture type or previously extracted region signature. We could invert this if we wanted a similarity measure. We could also weight the component single-band distances differently if we had knowledge that some data bands were more critical for recognition than others.

The easiest way to determine, or to "learn," which data bands are important is to keep track of the multivariate statistics within a target population and compare them with the statistics for other possible scene entities. This leads to Mahalanobis distance as a measure of dissimilarity between a pixel and a prototype. As discussed above, this would be optimal for Gaussian distributions, since they are fully characterized by their means and covariances.

It is not necessary to represent a texture prototype by statistical vectors and matrices, nor is it necessary to specify complex parsing rules. An intermediate strategy is to represent a texture or target signature by its full histogram in each data band. (A knowledge-based system would also have rules for manipulating these histograms in accordance with overall image illumination and contrast; we shall assume here that any required normalization has been done or will be compensated for during the image analysis.)

Storing histogram vectors as prototypes is very easy for a region-based system because the region histograms are always readily available. To train the system one has only to trace or extract a suitable region, assign it a label, and store it in the knowledge base. The only complication is that different data bands may be used during different image analyses, so that prototypes saved during one session may not include the bands needed during another session. There would also be some difficulty if the same data band were scaled or quantized differently for each session, but we can usually compensate for such discrepancies.

Computing similarity between a pixel and a prototype is a bit more difficult than computing Mahalanobis distance. We can split the problem into that of computing similarity within a given band and that of combining different band similarities into a single overall similarity measure, although such a two-step procedure is not necessarily optimal.

Consider then the problem of estimating how likely it is that an observed gray level in a data band came from one prototypical population and not from another. We may formalize and generalize this problem as follows. Given that we have observed a gray level g as an independent random sample, what is the probability that the source population was one of a set ω of positive exemplar distributions, ω_i, and not from one of a set ϕ of negative exemplar

distributions, ϕ_j? We shall assume that we know each of the prototype distributions by a single histogram representing an unbiased sample from some large population. We have, then, sets of histograms estimating $\Pr(g|\omega_i)$ and $\Pr(g|\phi_j)$ and we wish to compute $\Pr(\omega|g)$.

If we assume disjoint and exhaustive source populations, we can compute the probability that the observed sample gray level g came from a positive exemplar source population as the normalized sum of probabilities for the members of the set:

$$\Pr(\omega|g) = \frac{\sum_i \Pr(\omega_i|g)}{\sum_i \Pr(\omega_i|g) + \sum_j \Pr(\phi_j|g)} \quad .$$

The denominator should sum to one, given our assumptions, but this formula will work even if the probabilities are expressed relative to some larger set of disjoint source populations.

Bayes' rule applied to each component term in the summations gives us

$$\Pr(\omega|g) = \frac{\sum_i \Pr(g|\omega_i)\Pr(\omega_i)}{\sum_i \Pr(g|\omega_i)\Pr(\omega_i) + \sum_j \Pr(g|\phi_j)\Pr(\phi_j)} \quad ,$$

where a term $\Pr(g)$ has been canceled from the numerator and denominator. This is the theoretical form of the similarity function that we need. If we assume equal a priori probability for each source population, the formula simplifies to

$$\Pr(\omega|g) = \frac{\sum_i \Pr(g|\omega_i)}{\sum_i \Pr(g|\omega_i) + \sum_j \Pr(g|\phi_j)} \quad .$$

Although the current SLICE program makes this simplification, additional knowledge of the scene domain might provide a better set of weightings.

Now, how do we compute $\Pr(g|\omega_i)$? We could simply take the bin count for bin g in the ω_i histogram and divide it by the total number of counts in the histogram. This would have two undesirable effects: the estimated probability for adjacent bins could vary wildly because of sampling fluctuations or "picket-fence" quantization effects, and the similarity formula could not be evaluated for bins that happened to be empty in all prototype histograms.

If the histograms were samples from Gaussian distributions, we could use the sample mean and variance to estimate the true population bin probabilities for every gray level. Since we generally have mixture densities, this approach would require that every prototype texture histogram be decomposed into component Gaussians. While this is difficult, it could be done (at least approximately) either automatically or interactively at the time a texture prototype is entered into the system's knowledge base. We note that this is an optimal solution, but will now proceed to develop a much simpler heuristic approximation.

If a distribution is known to be Gaussian, we achieve the greatest predictive power by using techniques appropriate to that parametric form. If we have no knowledge of the parametric form, we can still treat the histogram as a sample taken from a multinomial distribution having unknown bin probabilities.

An observed bin probability, $\Pr(g|\omega_i)$, is an unbiased estimate of the true bin probability in the sampled population. It is not, however, the best estimate of that generating probability, given that a sample has been taken. An example may clarify this somewhat difficult concept. Suppose that we have formed our prototype histogram by sampling a single pixel. We shall then have a single populated bin and 255 empty bins (assuming 8-bit

quantization). Are we then willing to say that our best estimate for the population distribution is a spike at the observed gray level and zero probability of any other value? No, we would wish to be more conservative, even if we were not assuming an underlying Gaussian model.

Our intuition serves us well here. Bayes showed in 1763 that the a posteriori probability of a given multinomial bin-generating probability, given an observed histogram, has a beta distribution, which is a continuous distribution resembling a skewed binomial [Jaynes 83]. The mean, or expectation, of this distribution is [Abramowitz 64, p. 930]

$$\frac{\text{observed bin count} + 1}{\text{number of bins} + \text{number of samples}}$$

Using this as our $\Pr(g|\omega_i)$ has the effect of smoothing the population histogram estimate slightly by adding a fractional count to each bin. This permits us to compute our similarity measure even for bins that are to be empty in all prototype histograms.

The similarity function is now optimal for multinomial distributions, but not for Gaussian mixture densities. It fails to allow either for the strong correlation between nearby bin counts that is due to the component densities or for the exponential decrease in bin frequency as a gray level is chosen farther from any histogram peak. The first effect is particularly noticeable when the training data contain regularly spaced empty bins resulting from a sticky quantizer bit or from contrast stretching that introduced a picket-fence envelope. While we can imagine separating two textures by their differing picket-fence characteristics (i.e., by trivial differences in gray levels), this is not the type of behavior we want to build into our image segmenter.

The solution, short of actually finding the component Gaussian densities, is simply to smooth the histograms. The SLICE program uses a binomial kernel (this is the best discrete approximation to a Gaussian) with a standard deviation of 1, 3, or 5 pixels. Histogram counts are scaled by 1000 so that fractional bin counts can be represented; this scaling must be compensated for when computing the similarity transform.

The above smoothing extends each tail of a histogram peak for a dozen pixels or so, then drops to zero. The multinomial bin correction that is subsequently applied will lift this slightly above zero, but by an amount that does not vary with distance from the histogram peak. This causes undesirable behavior of the similarity transform for gray levels near the ends of typical histograms. Consider the case of a very sharp Gaussian peak for our positive exemplar and a broad peak or mixture density for our negative one. Further assume that the positive-exemplar histogram contains only a few hundred pixels and that our negative exemplar is based on a very large sample, typically the entire image we wish to segment. We would expect that image pixels far from the positive exemplar peak would have very low similarity to that texture type because the associated Gaussian distribution would have a very sharp exponential decay. Instead we compute a high similarity because the multinomial correction for a histogram with few counts is a much larger number than that for a histogram with many.

This leads to one more adjustment, a factor that provides exponential (i.e., Gaussian) decay in the multinomial correction as we select bins farther from the nearest or broadest peak in a histogram. There is no need to be precise here, so we can use an approximation based on the distance to the lowest or highest count in the smoothed histogram. Only the multinomial correction is applied for gray levels that are between these two limits. For

pixels outside the observed sample range, the 1 in the multinomial correction is replaced by $\exp(-\frac{2d}{w})$, where d is the distance to the nearest bin in the observed range and w is the number of bins in that range.

6. Goal-Directed Segmentation

We now have a rapid method of computing the similarity of an observed gray level in an image band to a set of positive exemplar textures (or target signatures) with respect to a set of negative exemplar textures. We can apply this function efficiently by precomputing the similarity measure for every possible texture-band gray level and then looking up each image pixel's value in the resulting table of values. We shall usually have a particular texture or target signature as a positive exemplar and shall use the full image (or region) histogram as a negative one. (This implicit inclusion of the texture we are seeking in the negative-exemplar histogram will not cause problems unless it is a major component of that histogram. If it is, we might first suppress that component of the negative-exemplar histogram by subtracting a multiple of the positive-exemplar histogram. The SLICE program does not yet include such a correction procedure.)

Our problem, then, is to select a similarity threshold that will separate all (or at least most) of the instances of our target texture from instances of all other textures. If multiple similarity bands are available, we should either select the best band for our threshold segmentation or combine the information in all the bands. This section describes a segmentation method capable of selecting the best similarity band for extracting examples of the target texture and of recognizing those cases in which no satisfactory threshold can be found. The next section will discuss other methods of combining information from multiple similarity bands.

For any texture band, the computed similarity value for each pixel should ideally be near 1.0 for the texture we are seeking and near 0.0 for the textures specified as negative training examples. The actual separation for any real data band will be less, and some texture bands may fail to discriminate the training textures at all, but a decision threshold at 0.5 should separate our positive and negative training textures if they are indeed discriminable. Decision thresholds above or below 0.5 could also be chosen; this is equivalent to adjusting the a priori source-class probabilities, $\Pr(\omega)$ and $\Pr(\phi)$, that are implicit in the similarity transform. (We can no longer second-guess the relative proportions of the population probabilities, $\Pr(\omega_i)$ and $\Pr(\phi_j)$, within the source class probabilities. Such fine control is not needed, however, particularly since we rarely require multiple positive or negative exemplars.)

"Unexpected" or unmodeled textures in the image should have similarity values in between the extremes for the training textures. The [eight-bit] histogram of a similarity band typically has a very large peak at the low-similarity end (representing image gray levels that were common in the negative-exemplar textures) and a spread of higher-similarity spikes that look rather uniformly distributed. Smoothing the histogram (with a Gaussian kernel of standard deviation 5 or less) typically reveals that this high-similarity energy consists of a few Gaussian clusters representing image regions that are fairly similar to the positive exemplar.

Segmenting this smoothed histogram is usually quite easy. The SLICE program cur-

rently uses a version of the PHOENIX histogram-based segmentation algorithm (documented in Appendix A) to find suitable thresholds. We may want to select just the peak of values most similar to the training texture, although thresholding anywhere above the large peak of least-similar values will generally produce a good spatial segmentation of the image. We may also select multiple thresholds that will isolate other peaks in the histogram and thus extract image regions with other textures as well. It is this capability that makes SLICE a segmentation algorithm rather than just a classification algorithm.

The above procedure works if it is possible to find even one peak in one histogram that is reasonably well separated from other peaks. In practice, there can be as much as a 25% overlap between two Gaussian peaks and the segmenter will still find reasonable subregions in the image. There will be times, of course, when the histogram-segmentation algorithm fails to find any segmentable peaks in the similarity-band histogram, particularly when we are trying to segment a whole black-and-white image or a low-resolution texture data band. The segmenter will find that a region is uniform and unsegmentable, but higher-level knowledge may suggest that this is false. The current SLICE program is not able to proceed automatically in such cases, but any of the following techniques could be invoked:

- Try again with relaxed parameters for the peak-finding heuristics. The SLICE program currently uses the PHOENIX histogram-partitioning heuristics with the "moderate" parameter settings developed during the SRI evaluation of that package for the DARPA/DMA Image Understanding Testbed. These smoothing parameters and heuristic criteria could be successively weakened until peaks are found in the histogram.

- Compute additional data-band transformations such as pairwise ratios or combinations of existing data bands. Any oblique cut through the multidimensional histogram space is likely to resolve at least one histogram peak. Computation of such a data band and histogram does not take long, particularly if only a small region is involved.

- Compute a multidimensional histogram from multiple data bands and apply cluster analysis techniques to find discriminable subpopulations. Combining two bands in this way produces a two-dimensional histogram that can be analyzed by means of image partitioning techniques [Nagin 77, 78]. The SLICE system itself might be used to find populated areas of the bivariate histogram.

- Threshold the image region at arbitrary levels, e.g., at histogram quartiles or deciles, and use spatial analysis (including noise suppression) to recover subregions that can later be remerged or edited. This option is available in the SLICE program and works surprisingly well.

- Partition the region at arbitrary spatial boundaries, segment the pieces, and then remerge or edit subregions along the boundaries [Robertson 73, Horowitz 74, Price 76].

- Switch to a different histogram-segmentation method, such as minimal-spanning-tree analysis or relaxation-based peak sharpening [Bhanu 82].

- Switch to an entirely different segmentation approach, e.g., region growing from homogeneous seed areas within the region.

Any of these methods, and no doubt others, will move the segmenter off "dead center" when segmenting a complex region is imperative. Once partitioning is started, subregions that are themselves composite can usually be segmented with ease.

Once distinct histogram peaks have been found, the segmentation algorithm finds corresponding regions in the image by simple threshold segmentation and connected-component extraction. It then computes a quality score for the spatial segmentation based on the percentage of small "noise regions" produced. This quality score can be used to select the best of several competing similarity-band segmentations. Better quality scores could be computed from region shapes and other high-level or goal-directed criteria.

The current SLICE program includes an optional screening of extracted regions based on spatial adjacency. Frequently the user wishes to "grow" an identified image region (e.g., one that he has traced with a pointing device) instead of finding all other similar pixel patches in the image. After connected-component extraction, the SLICE program can suppress any region that is not touching or nearly touching the initial prototype. Connected components are again extracted and the analysis proceeds. The final step in the growing process is to merge the regions found with the original training region.

7. Multiband Similarity: Putting It All Together

The previous section described a method for finding image regions corresponding to peaks in a similarity-band histogram. The implemented segmentation algorithm is able to select the best of several competing similarity bands by comparing the identified histogram peaks and quality of spatial segmentation produced by each set of similarity-band thresholds. This approach is typically useful in cueing applications when searching a scene for textures that might differ considerably from stored prototypes. Using this technique, a target region distinguishable in even one data band can be segmented from its background and passed up to a higher-level reasoning system for confirmation. The method also works well with multiple data bands containing essentially the same information, since slight differences in the information content might lead to better segmentation in one band than in the others.

Another approach, also available in the current SLICE program, is to combine the similarity bands computed from different luminance or texture bands into a single overall similarity band. This is appropriate when we are very sure that our prototypes are representative, as when we are trying to find a homogeneous texture region around a traced seed region. Under these conditions we can assume that all instances of the target texture will look very similar to the training texture in all transformed bands—if a subregion differs significantly in even a single band it cannot be from the target population.

We might use factor analysis or discriminant analysis to devise an optimum weighting function for combining the similarity bands. Such a function would no doubt be task-dependent and image-dependent, making it very difficult to assemble sufficient training data. A simpler solution is to construct the composite similarity band from the pixel-by-pixel minima of the component similarities. This combining function is often used in fuzzy-set theory. It correctly reflects the assumption that target textures should behave just like the prototype texture under any transformation, but has the negative effect that we cannot recover from a prototype that is unrepresentative in even a single data band.

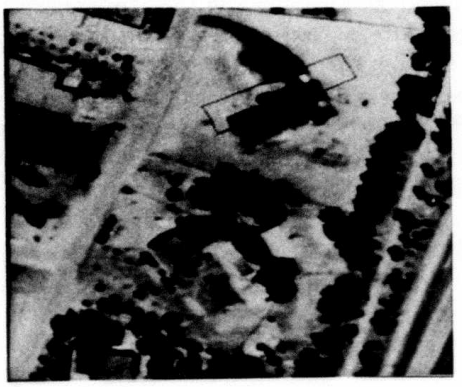


Figure 2: Aerial Image with Positive and Negative Training Regions

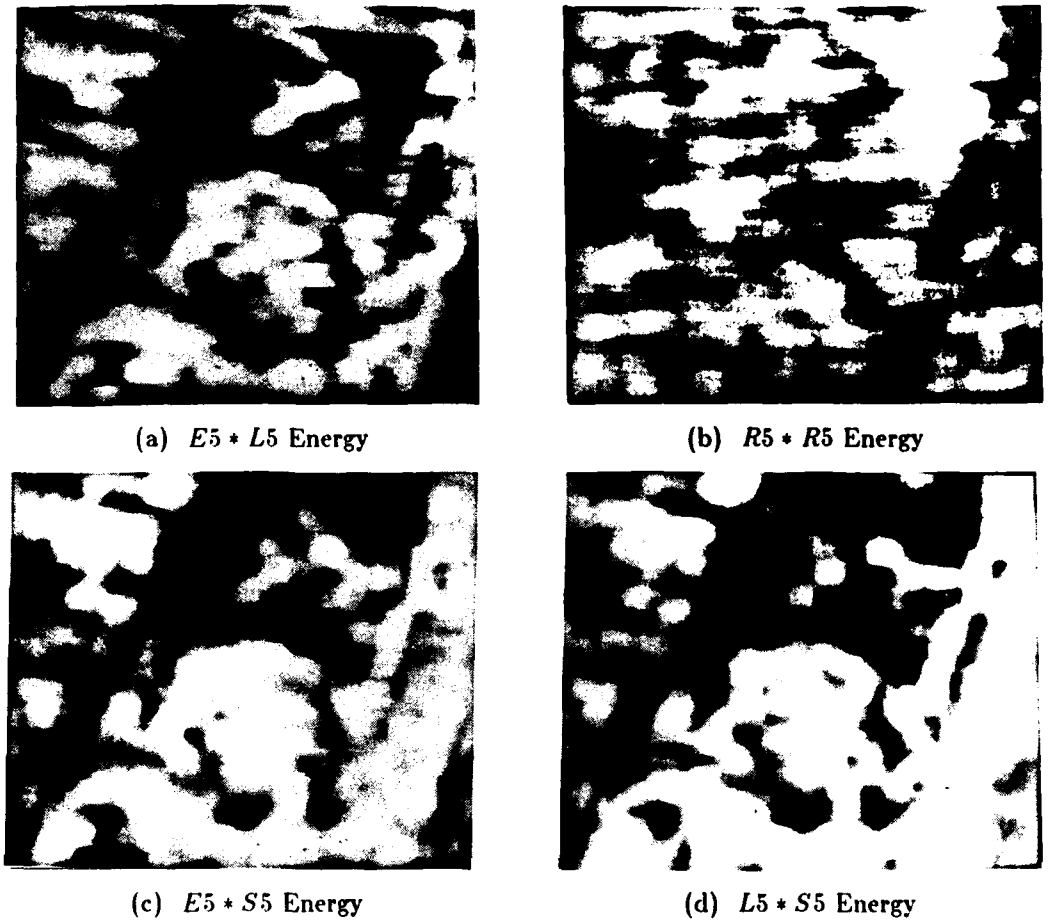
In practice, the use of similarity minima works quite well for region growing. When combined with spatial analysis and noise cleaning it results in either a reasonable segmentation or a failure to segment at all. Other combining functions might work better in particular cases, however. Higher-level guidance based on preliminary segmentation and analysis of each similarity band could be used to choose combining functions intermediate between the first all-or-nothing approach and the second pixel-by-pixel minima approach [Salton 83, Rauch 84].

One of the consequences of the SLICE similarity band computation is that the inclusion of additional similarity bands in a composite should never degrade performance (other than taking longer to compute). If we have truly representative prototypes, any histogram-based transformation will either help discriminate the positive training classes from the negative training class or it will fail to do so. If it discriminates them partially, any reasonable combining function will either make use of the information or, at worst, ignore it; if it does not discriminate them, the corresponding similarity band will be essentially constant and will do no harm. (The SLICE program currently tests for such useless similarity transforms and does not bother to compute the similarity band.) The SLICE similarity-transform approach thus has the advantage that it may fail to provide information but will seldom produce misinformation.

8. Examples: The Proof of the Pudding

We shall now examine the performance of the SLICE algorithm on a particular aerial image analysis task. The processing sequence will demonstrate both the advantages and the disadvantages of this approach. Simple ways of improving the demonstrated performance will also be discussed.

Figure 2 is a black-and-white image of a residential area near Page Mill Road in Palo Alto, California. The image has not been normalized or otherwise preprocessed. It shows trees, roads, fields, buildings, swimming pools, and a few cars. Shadows are cast both by the buildings and by the trees, although the resolution is such that tree shadows are very



(a) $E5 * L5$ Energy

(b) $R5 * R5$ Energy

(c) $E5 * S5$ Energy

(d) $L5 * S5$ Energy

Figure 3: Texture Energy Bands

difficult to discern. A full parse of the scene would detect and identify all of these entities. Our concern here will be the extraction of all the tree regions, perhaps as a preliminary to extracting other objects in the scene. We shall first trace the process of "growing" tree regions from user-selected examples and shall then examine the output of a more general "finding" or cueing algorithm.

Extraction of the trees in this image by interactive threshold segmentation is not difficult. The trees (or perhaps bushes) are distinguishable by their gray-level signatures in the original image. An image-understanding system would not know this a priori, however, but would have to extract and identify at least some of the trees and then estimate whether it could extract the rest. The system could search for good tree regions by experimentation with different thresholds [Selfridge 82], but the methods presented in this paper are more efficient.

Trees presumably have distinctive texture signatures in addition to their gray-level signatures. Figure 3 shows four texture bands selected from the texture energy set. These were computed with 5×5 filters masks and 15×15 "absolute average" summations. (At this

image resolution, the 3×3 filter set would probably have worked better.) Each texture band highlights different characteristics of the original image. The four texture bands used here are the same ones selected in the development of the texture energy measures [Laws 80], although there is little reason to believe that this is the best subset or even an adequate subset for image analysis. The selected filters respond primarily to horizontal edges, high-frequency variation, medium-frequency diagonal structure, and narrow or high-frequency vertical structure. The data bands have been normalized for image contrast, but pairs have not been combined to form rotation-independent texture bands.

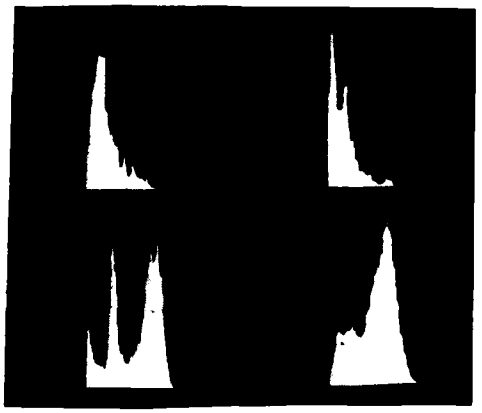
Goal-independent analysis of the original image, using the PHOENIX segmentation algorithm (and skillful setting of its numerous parameters), leads to isolation of most trees in the scene. These regions are not the first to be reported by the segmenter, however, nor are they the last. The user or high-level control program must somehow select the tree regions from among the hundreds of reported regions. This is made more difficult by the fact that most of the scene is very poorly segmented by the PHOENIX algorithm, with region boundaries crossing homogeneous fields and with parts of house roofs cut off and grouped with surrounding fields. Adding computed texture bands to the original black-and-white band degrades performance: areas around building edges are identified as homogeneous regions and several scattered patterns of trees interspersed with grass are also extracted as regions. (The latter effect is useful, but not as useful here as finding the individual trees.)

Goal-driven segmentation with the SLICE algorithm begins with the selection of training areas. A sophisticated system might have adequate tree templates stored in its knowledge base. Here we depend on the user to select representative training regions. Large samples work better than small ones, but we will demonstrate the technique with the two small training areas in Figure 2. A negative training region containing a strong shadow and a mixture of other scene textures is also shown; the remainder of the image outside the three traced areas will be used as a second negative training region.

The two positive training regions were carefully selected. The upper-right region is a nearly minimal sample such that the SLICE program's "grow" command will extract the entire clump of trees extending from the upper-right corner diagonally downward toward the bottom edge of the second training region. The second region is also a nearly minimal sample such that the "grow" command will extract all tree clumps touching the region. Rather than witness these feats, we will now examine performance when both regions are sought simultaneously and the upper negative training example is also specified.

Figure 4(a) shows the histograms of the training regions together with the histogram of the remainder of the image. We can see that both tree regions have similar histograms, although they differ in detail. The negative training region has three peaks corresponding to a shadow, the house roof plus driveway, and the lawn and car. The histogram for the rest of the image contains some dark pixels from trees and shadows and many light pixels from other scene components.

Figure 4(b) shows the resultant gray-level similarity transform for this black-and-white image band. Similarity is highest for those pixels corresponding to trees, despite the negative effect of tree regions in the whole-image histogram. The similarity transform shows a very slight dip for shadow pixels and a much stronger dip in the house roof interval. The least-similar gray levels are those occurring in the image but not in either positive training region. The similarity function increases again for very bright pixels: these are absent in all image

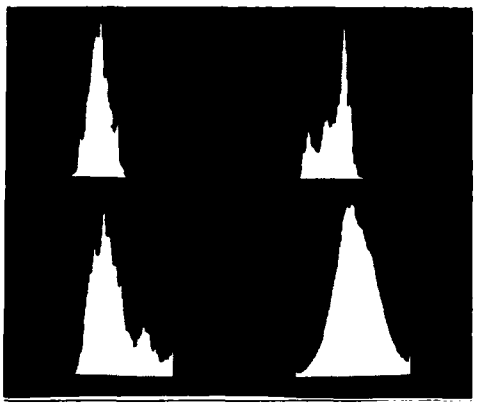


(a) Training Histograms

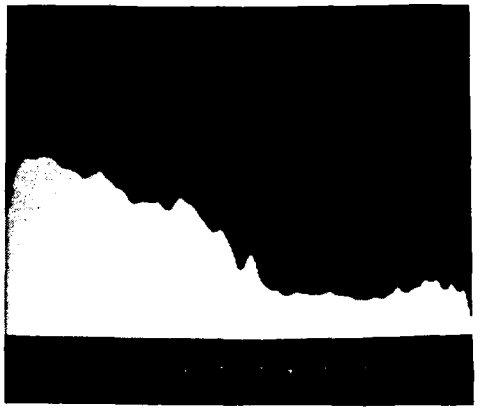
(b) Similarity Transformation

(c) Similarity Band

Figure 4: Original Image Analysis



(a) Training Histograms



(b) Similarity Transformation

(c) Similarity Band

Figure 5: $R5 * R5$ Texture Band Analysis

regions and hence need not concern us.

Applying this transformation to the original image produces the similarity data band in Figure 4(c). Trees, shadows, and a very small number of other scene objects have been highlighted. We can easily extract trees from this transformed band, and such a segmentation will be presented in Figure 8. For now, let us continue trying to "grow" our training regions using multiband similarity.

The SLICE program is capable of using any number of texture bands as additional inputs. For demonstration purposes, we shall limit it to just the band of Figure 3(b). This band tends to show trees as dark regions, although the effect is not strong. Any of the other texture bands might perform as well or better.

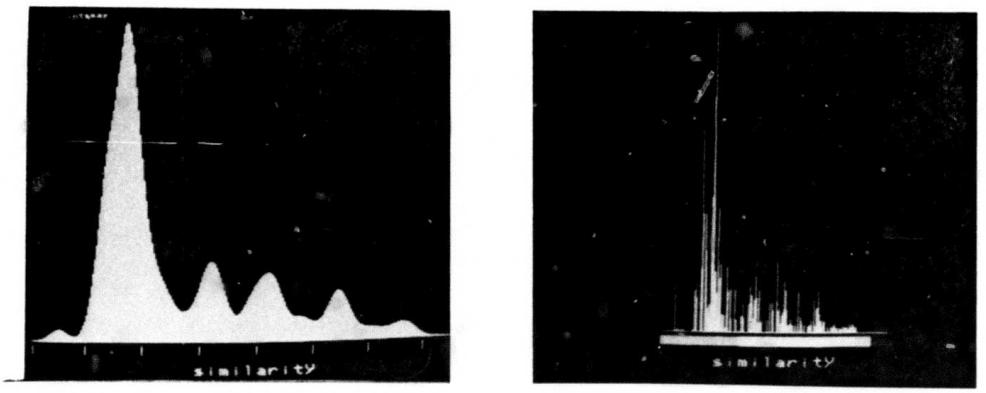
Figure 5(a) shows the training histograms for this band. Note how strongly the negative training area histogram matches that of the positive training areas. Use of the negative training example actually degrades segmenter performance in this case, albeit only slightly. (In other situations the "caution" introduced by this overlap might prevent the segmenter from making bad decisions.) We could reduce the degradation by giving the negative example less weight than the histogram of the area to be segmented, but the knowledge-based mechanisms needed to make such decisions have not been included in the SLICE program.

Figure 5(b) shows the similarity transform computed from the training histogram. It shows a preference for dark pixels, but is not very specific. This leads to the texture-similarity band of Figure 5(c), which we can see will not lead to a good segmentation of the image. The segmentation program has no such perception, however, and must somehow determine that it should reject most of the "information" in this computed band.

The method of combining similarity functions that we will use here is to take the pixel-by-pixel minimum of all similarity bands. This combined transform function can be computed from the individual similarity transformations rather than from the similarity data bands, thus saving considerable computation. The result of applying this combined similarity transformation to the original image may be seen in Figure 7(a). It is similar to the similarity band for the black-and-white data band, although more intermediate gray levels are present.

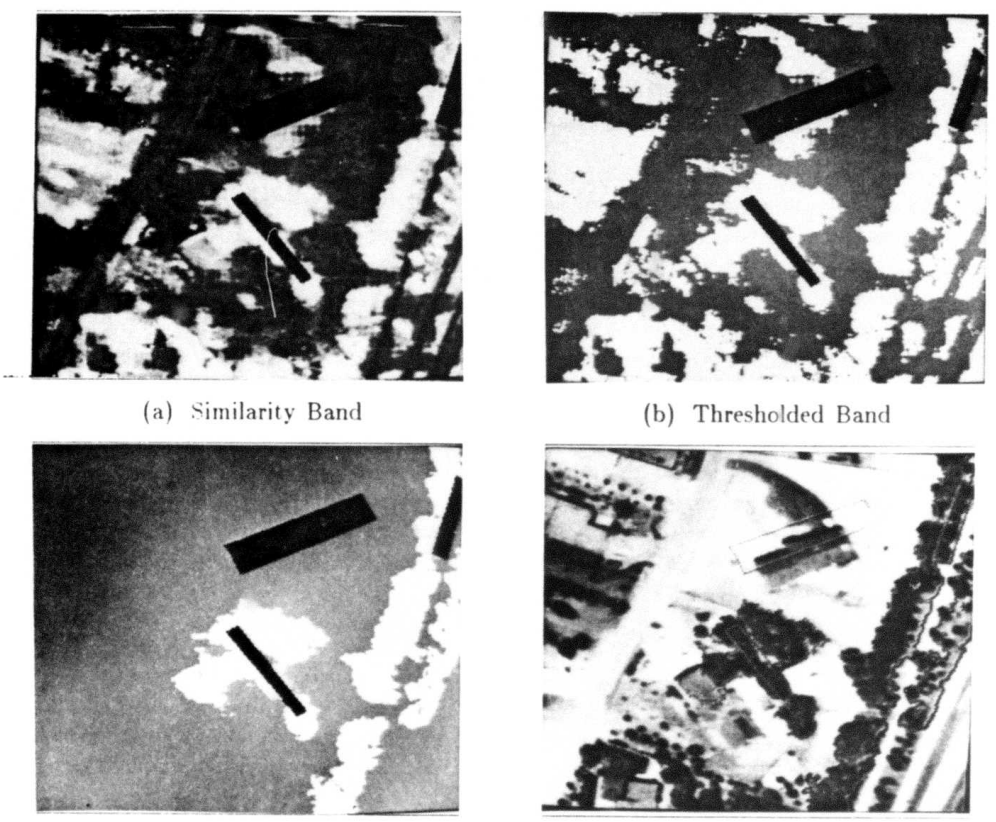
Figure 6 shows both the smoothed and the unsmoothed histograms of the merged similarity band. There are several clusters in this one-dimensional space, and any to the right of the main histogram peak could represent the trees we are seeking. An intelligent system would investigate several thresholds or would use several thresholds simultaneously. The current SLICE algorithm simply chooses the threshold that best survives its screening heuristics—in this case perhaps a rather poor choice. Figure 7(b) shows the spatial result of applying this threshold. The trees are indeed found, but so are driveways, road patches, and other dark image regions. There has also been a blurring effect because of the 15×15 window used to compute the texture energy band.

Our current task is to "grow" the original sample regions to their full image extent, rather than find more distant matching regions. A simple spatial analysis can thus be employed to eliminate all regions not touching the original training regions. (The current criterion is that the rectangle enclosing a candidate region must come within one pixel of touching a rectangle enclosing one of the positive training regions. This allows for small breaks in our extraction of a scene object and permits a higher-level process to determine



(a) Smoothed (b) Unsmoothed

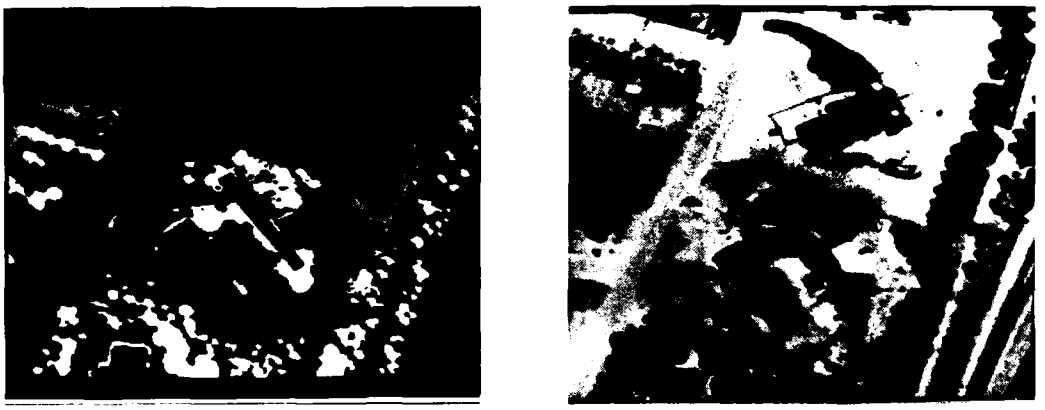
Figure 6: Merged Similarity-Band Histogram



(a) Similarity Band (b) Thresholded Band

(c) Extracted Regions (d) Region Outlines

Figure 7: Merged-Similarity-Band Analysis



(a) Gray-Level-Coded Region Map (b) Region Outlines

Figure 8: Regions Found by Similarity-Band Selection

whether nearby regions should be merged or discarded.) Candidate regions smaller than some threshold, here set at five pixels, are also discarded as probable noise regions.

Figures 7(c) and 7(d) show the result of this spatial screening. The segmenter has done a good job of expanding the initial sample regions, although the lower-left region has absorbed part of an adjacent house roof. The upper-right tree clump has been found, but includes pixels from the surrounding field that would not have been included if the inherently blurred texture band had not been used (or if the final similarity histogram had been thresholded at a higher gray level). Two additional large regions, both containing trees, are found because of the gray-level and spatial interactions of the two sample regions. They would not have been retained if the two training regions had been grown independently.

This concludes the presentation of the goal-directed region-growing technique used in the SLICE program. We have seen how the algorithm is able to overcome difficulties such as small positive training regions; negative training regions that are unrepresentative (i.e., badly weighted) and include the very pixels we are trying to find; blurred, poorly chosen, or uninformative texture data bands; ad hoc similarity combining functions; and poorly chosen thresholds. As knowledge-based techniques are developed and refined, some of these difficulties will be eliminated and performance enhanced commensurately.

A final example of the power of the SLICE goal-directed approach may be observed in Figure 8, which shows the effect of the SLICE program's "find" command when the same training regions and texture band are employed. This algorithm differs from region growing only in the similarity combining function and spatial screening. Instead of combining the computed similarity bands, each is analyzed separately and the one with the least "noise area" is selected. The texture data band is thus rejected and only the original black-and-white image is employed. Shadows and a few other undesired dark areas are found, but essentially every tree over five pixels in area is identified. This leaves very little work for a higher-level verification process to perform.

9. Conclusions

The SLICE segmentation system is one of several existing systems for segmenting digital images recursively. Its major contributions are computation of a nearly optimal texture-similarity function and integration of this approach with a robust segmentation system to permit both goal-directed and goal-independent textured-image partitioning. Some of the advantages and disadvantages of the SLICE algorithm are listed below.

- The SLICE goal-directed segmentation algorithm uses multispectral or "multitextural" input to extract precisely those scene objects of most interest. If it fails, repeated attempts with relaxed constraints may locate candidate regions. If it succeeds, it generally produces high-quality regions that require little postediting. Provision for negative training examples permits easily confused material types to be separated early in the analysis process.

- SLICE, like other region-based methods, always yields closed region boundaries. This is not true of edge-based feature extraction methods, with the possible exception of boundary following and zero-crossing detection. Closed boundaries, the essence of segmentation, greatly simplify other image analysis tasks such as material identification and object mensuration. The resulting regions provide meaningful entities for a human or high-level control system to reason about and manipulate.

- SLICE is a hierarchical or recursive segmenter, which means that even a partial segmentation may be useful. This can save a great deal of computation if efforts are concentrated on image regions in which further segmentation is critical. If a full goal-independent segmentation is desired, however, other methods of segmenting may be more economical.

- SLICE is relatively insensitive to noise because noise tends to average out in the region histograms used to select thresholds. This contrasts with edge-based methods, as the local analysis they require can be highly perturbed by noise.

- SLICE currently has no notion of boundary straightness or smoothness. This may be either good or bad, depending on the scene characteristics and the analysis task. It easily extracts large homogeneous regions that may be adjacent to detailed, irregular regions (e.g., a lake adjacent to a dock area or the sky over a complex skyline); such tasks can be difficult for edge-based segmenters. Boundary aesthetics and other semantic criteria can be incorporated as part of either an editing process or knowledge-based control structure.

- Region-based segmenters may fail to detect even long and highly visible boundaries between two large, similar regions if the region textures cause their histograms to overlap. The use of texture bands reduces this problem because the boundary region itself forms a distinctive texture. Hypothesis-driven edge-based methods may be required to confirm such boundaries.

- SLICE tends to miss small regions within large ones because they contribute so little to the composite histogram. It is thus poorly suited to goal-independent detection

of vehicles and small buildings in aerial scenes, although the use of multiple texture bands alleviates this deficiency. Goal-dependent selection of search areas and texture-similarity transforms will help to locate small objects even against backgrounds of similar gray level.

- SLICE also tends to misplace the boundary between a large region and a small one, thus obscuring roads, rivers, and other thin regions. Boundaries found by edge-based methods are less affected by distant scene properties, but work poorly if not adapted to the statistics of the regions being discriminated. An edge-based postediting of the region boundaries found by SLICE may combine the best of both approaches.

- SLICE requires multispectral input or multiple texture transforms for effective operation. Edge-based and valley-seeking or spanning-tree techniques are better adapted to operation in a single data band, and thus require less computer memory and possibly less processing time.

Selection of a segmentation algorithm should depend on the task to be performed. The SLICE segmentation system is a convenient testbed for integrating diverse feature-extraction techniques and experimenting with knowledge-based control structures.

Appendices

A. The PHOENIX Segmentation Algorithm

The SLICE segmentation algorithm incorporates the PHOENIX color segmentation algorithm developed at Carnegie-Mellon University [Shafer 82, Laws 82]. This is a sophisticated method of hierarchical region extraction based on region statistics and user-specified parameters. It does not use explicit knowledge about the types of data bands it is given nor about the scene objects being sought.

Each object or object part in a scene is assumed to form a nearly uniform patch in the image, with a noisy Gaussian peak in any single-band histogram. Decomposing a function into Gaussian peaks is known as the mixture density problem [Wolfe 70] and is important in information theory, statistics, chemistry, and other fields. Very little of this theory has been applied to image processing [Chow 70, Rosenfeld 76, Postaire 81]. The PHOENIX/SLICE algorithm segments mixture densities by identifying the most obvious thresholds in any of the data bands, then using spatial-analysis "look-ahead" before confirming a candidate threshold. The algorithm does make slight errors in threshold placement, however, leading to the breakup of some small regions and a shifting of the boundaries of others.

Ohlander and Price used a hierarchy of heuristic rules for selecting the most prominent peak within a set of histograms [Ohlander 78, Price 79, Nevatia 82]. PHOENIX uses similar heuristics, but concentrates on the valleys (i.e., local minima) in the histogram set. Usually a single valley, resulting in one threshold and two intervals, is selected for each feature. Spatial analysis is then employed to select the best threshold/data band combination. Using only one threshold per pass reduces the chance of segmentation errors, although it does increase the number of passes required.

Histograms can be treated as one-dimensional images and can be segmented by almost any image segmentation method. The PHOENIX histogram-analysis component uses an *interval-merging* strategy. Each single-band histogram is first smoothed with a binomial or Gaussian smoothing kernel having a standard deviation **gsmooth**— typically 3, and ranging from 5 for coarse segmentation down to 2 for more detailed segmentation. (The original PHOENIX algorithm used a simpler unweighted moving average.) The histogram is then broken into intervals in such a manner that each begins just to the right of a valley (i.e., at the next higher intensity), contains a peak, and ends with the next valley. A valley is considered to be the right *shoulder* of its left interval and the left shoulder of its right interval. The leftmost and rightmost intervals always have exterior shoulders of zero height.

A series of heuristics is then applied to screen out noise peaks. Each test is applied to all the intervals in the histogram. When an interval is eliminated, it is merged with the neighbor sharing the higher of its two shoulders. The screening test is then applied again to the merged interval. (Previous tests are not reapplied.)

Peak-to-shoulder ratio is tested first. An interval is retained only if the ratio of peak height to the higher of its two shoulders, expressed as a percentage, is at least as great as the user-supplied **maxmin** parameter—typically 160%, and ranging from 300% for "strict" screening down to 130% for "mild" screening.

Peak area is then compared with an absolute threshold, **absarea,** and with a relative threshold, **relarea**, representing a percentage of the total histogram (or region) area. Only peaks larger than these thresholds are retained. **Absarea** is typically 30 pixels, ranging from 100 pixels down to 5 pixels; **relarea** is typically 2%, ranging from 10% down to 1%.

The intervals surviving to this point should be reasonable candidates, and it is fairly safe to use global histogram descriptors in the test conditions. The second-highest peak is now found, and those peaks whose height is less than a percentage, **height**, of it are merged. The lowest interior valley is then found, and any interval whose right shoulder is more than **absmin** times that valley height is merged with its right neighbor. (The parameter appears to be misnamed, since the criterion is relative rather than absolute.) Typical values of these parameters are 20% and 10 pixel counts, ranging from 50% to 10% and 2 counts to 30 counts.

A final screening is performed to reduce the interval set to **intsmax** intervals. This is done by repeatedly merging regions with low peak-to-shoulder ratios until only **intsmax** -1 valleys remain. **Intsmax** is typically set to 2 to force the highest-quality segmentation during each pass, although higher values could save considerable computation time.

A score is also computed for each interval set as a whole (in relation to the interval sets for other data bands). This score is the maximum over all intervals of the function

$$1000\,\frac{\text{peak height} - \text{higher shoulder}}{\text{peak height}}$$

This formula assigns the maximum score to an interval set containing a peak with shoulders of zero height. Interval sets with scores less than **absscore** or less than **relscore** percent of the best score for all data bands are rejected. **Absscore** is typically 700, ranging from 925 down to 600; **relscore** is typically 80%, ranging from 95% down to 65%.

If more than **isetsmax** data bands are still candidates for segmentation, the excess ones with the lowest scores are now dropped. This parameter is typically 3 and ranges from 2 to 5. Remaining data bands and interval sets are passed to the spatial-analysis subsystem.

Histogram segmentation is a heuristic technique that sometimes misses good thresholds and sometimes chooses bad ones. Some protection is provided by examining segmentations of several different data bands and choosing the best. Regions smaller than the **noise** threshold are merged back into their parent regions and bands producing region segmentations with more than **retain** percent of their area so merged are rejected. These parameters are typically 10 pixels ranging from 50 pixels down to 5 pixels, and and 20% ranging from 4% to 40%. The remaining segmentation producing the lowest noise percentage is then selected and instantiated in the data base. All resulting subregions are scheduled for further attempted segmentation provided that their areas are at least **splitmin** pixels—typically 40 pixels and ranging from 200 pixels down to 20 pixels.

No single threshold is going to result in perfect segmentation when the histogram peaks overlap. We might instead use two thresholds—one low enough to catch all of the higher peak and another high enough to catch all of the lower peak—then ascertain from the image which threshold is correct for extracting each subregion. In practice, most of the small *noise patches* that result from a slightly offset threshold are easy to identify and absorb into the surrounding subregions. The noise-cleaning process leaves only the exact placements of the subregion boundaries in doubt, and these can be better determined in a postediting of adjacent region pairs than through clever partioning of a multiregional histogram.

B. Spectral Transforms for Color Segmentation

Color bands are needed when two regions to be distinguished have similar texture (including intensity), but different hue or saturation. Transformations of these bands can sometimes be used to separate pixel clusters that project to overlapping or confounded histogram peaks in the original spectral data bands. Similar band combinations may be useful for segmenting texture bands or other nonspectral data bands.

Color transformations are not currently implemented as part of the SLICE program, but transformed data bands computed off-line can be used to improve its operation. The segmentation algorithm currently makes no distinction between color bands and other types of data bands, although the associated display routines do make such a distinction.

Color bands for image processing research are typically generated by scanning a color photograph through filters (e.g., Wratten filters 25, 47B, and 58) to get red, green, and blue (RGB) data bands. Real-time systems often use an electronic color camera to generate equivalent YIQ[3] bands that correspond roughly to perceptual brightness, cyan vs. orange, and magenta vs. green. The following discussion assumes that the primary input is in RGB coordinates, but converting to or from other color coordinates is fairly easy.

Each color system constitutes a three-dimensional chromatic space that can express most of the colors perceived by humans. (The detailed spectrum that astronomers and other physical scientists depend upon has been lost, just as it is in the human visual system.) A few purples and highly saturated colors are not precisely representable and the colors recorded with different films or cameras may differ, but the tricomponent representation is adequate for most purposes.

[3] YIQ is the National Television Systems Committee (NTSC) color coordinate system. The perceptual brightness, or Y, chromaticity band takes its name from the XYZ chromatic primary system of the Commission Internationale de l'Eclairage. I and Q are the NTSC *in-phase* and *quadrature* signal components.

Typical quantization is eight bits per color axis, or 16.8 million cells for an entire three-dimensional color histogram. Cluster analysis in such a space is not attractive, although methods of multidimensional pattern recognition are available. The SLICE package instead uses an adaptation of a one-dimensional histogram partitioning method implemented in the CMU PHOENIX program [Tomita 73, Tsuji 73, Ohlander 75, Shafer 82, Laws 82].

Any one-dimensional histogram is equivalent to a projection of the three-dimensional data onto a line (or curve) through the chromatic space. If the scene contains many regions, their histogram peaks are likely to overlap and obscure any useful details in the composite histogram. The overlap is different for projections at different angles, and it is often possible to isolate peaks from some of the regions by using many projections.

Ohlander used RGB, HSD[4], and YIQ projections, but many other color coordinate transformations are possible. The HSD coordinates were introduced by Tenenbaum et al. [Tenenbaum 74a, 74b] to mimic human color perception. They are

$$H = \arccos \frac{(R-G)+(R-B)}{2\sqrt{(R-G)(R-G)+(R-B)(G-B)}}$$

$$S = m \cdot (1 - 3\frac{\min(R,G,B)}{R+G+B})$$

$$D = \frac{(R+G+B)}{3} \quad ,$$

where m is the maximum desired saturation value. Hue is normalized by subtracting it from 2π if $B > G$; some care must be taken in rounding the values near 2π if the number is quantized. Note that these formulas contain singularities that are due to division by zero, and thus exhibit unstable segmentation behavior near the D axis.

The YIQ coordinates used in color television transmission are

$$\begin{aligned} Y &= 0.509R + 1.000G + 0.194B \\ I &= 1.000R - 0.460G - 0.540B + M \\ Q &= 0.403R - 1.000G + 0.597B + M \quad , \end{aligned}$$

where M is the highest possible intensity value in the original RGB features, typically 255. These formulas have been linearly scaled to maintain quantization accuracy (via the unit coefficient). M is added simply for convenience in digital representation. (The Q feature can be negated before adding M to better match the green gun on a color monitor.)

Kender analyzed the color transformations used by Tenenbaum and Ohlander and showed that inherent singularities and quantization effects were capable of introducing false histogram peaks and valleys [Kender 76, 77]. This effect is particularly noticeable in the hue feature, but also affects saturation and other normalized chromaticity coordinates. He recommended that saturation be ignored in regions of low luminance, with hue ignored in low saturation as well. The YIQ transform was found to entail fewer problems, although its usefulness in segmentation was not evaluated. Kender also proposed an improved computational algorithm for hue.

[4]The HSD, or hue-saturation-intensity, color coordinate system is also known as the HSI or IHS system. The symbol D is used here for intensity to avoid confusion with the YIQ system. It comes from *density*, a measure of the amount of silver deposited at a given point in a photographic negative.

Ohta et al. have further investigated color transforms for recursive segmentation [Ohta 80a, 80b]. They computed color histograms by using the Karhunen-Loeve color transform—an expensive method because the transform is different for each region. Ohta found that the principal transform axes typically clustered around

$$\begin{aligned} I_1 &= R+G+B \\ I_2 &= R-B \\ I_3 &= 2R-(G+B) \end{aligned}$$

and recommended that these features be employed. (The second and third features may be negative, so that either an offset becomes necessary or the segmentation code must be able to handle negative pixel values.)

Ohta's transform is similar to the YIQ system, as well as to the "opponent" color process recommended by several authors [Sloan 75, Nagin 78]. The transform is linear and hence avoids the instabilities that Kender found in saturation, hue, and normalized chromaticity coordinates. Nagin expressed some theoretical reservations about his own opponent features, but concluded that they "consistently provided more discrimination than the original RGB data."

HSD and YIQ color transformations were used in SRI's evaluation of the PHOENIX color segmentation program [Laws 82]. Hue was mapped to the range 0 to 179, with red at 0 (and 180), green at 60, and blue at 120. Achromatic pixels (i.e., black, gray, and white) were mapped to 255; this seldom makes a difference since pixels with exactly equal RGB components are exceedingly rare. A less exact test for achromaticity might work better (or at least differently) for images with slight imbalances in their color strengths.

The I and Q color bands computed by Kender's formulas should theoretically be divided by two (and then shifted to a nonnegative range) if they are to be stored in 8-bit image planes. (The SLICE program can handle image data with other pixel sizes, but eight bits is convenient and seems to offer a reasonable dynamic range.) Most of this range is wasted, however, unless I is stretched by a factor of two and Q by a factor of four prior to quantization, with clipping of extreme values. This greatly increases the usefulness of these bands for segmenting natural imagery, although it could fail for scenes that contain large regions of saturated colors.

Hue was generally not only the most useful color band in the SRI evaluation, but also the easiest to comprehend. The D and Y bands are essentially redundant; they do not always segment identically, but the extra information is not worth the effort of computing both. Segmentation on the RGB bands was almost as good but more difficult to explain; the RGB bands were each nearly equivalent in segmenting power, and successive region extractions seemed to jump randomly from one to another. (Differences in color are usually correlated with disparities in brightness, so an object that appears red might actually be segmented on a different color band by the PHOENIX/SLICE algorithm.) The S band was somewhat less useful, although decisions based on it were easy to explain. I and Q were the least useful data bands, although they might have been essential if some of the other seven data bands had not been available.

Overall, the HSI color system seems easiest to use, although the other color systems work well if explanations of each segmentation step are not needed. The RGB bands are so similar to one another that the addition of a hue band can improve segmentation greatly.

Pixels containing blue mixed with red (i.e., purples and violets) are rare even in hazy mountain scenes, so there is seldom a problem with peaks in the hue histogram being split between the bottom and top portions of the scale. Saturation is more likely to be the cause of such instabilities; dark or shadowed image regions sometimes transform to very high saturation values, indicating that segmentation on luminance should be done first or that the instability of saturation should be considered during noise cleaning and other analyses.

Transformations of texture and other nonspectral data bands have not been evaluated, but simple sums and differences of the bands are the most separated in a multidimensional histogram space and are thus most likely to improve segmenter performance. Texture bands can also be combined with spectral bands in this way.

References

[Abramowitz 64] M. Abramowitz and I.A. Segun (eds.), *Handbook of Mathematical Functions*, National Bureau of Standards (1964). Reprinted by Dover Publications, Inc., New York, New York (1965).

[Bhanu 82] B. Bhanu and O.D. Faugeras, "Segmentation of Images Having Unimodal Distributions," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 4, pp. 408–419 (July 1972).

[Chow 70] C.K. Chow and T. Kaneko, *Boundary Detection of Radiographic Images by a Threshold Method*, IBM Research Report RC-3203 (1970). Also in *Proc. IFIP 71*, TA-7, p. 130 (1971), and in S. Watanabe (ed.), *Frontiers of Pattern Recognition*, Academic Press, New York, New York, pp. 61–82 (1972).

[Harwood 83] D. Harwood, M. Subbarao, and L.S. Davis, *Texture Classification by Local Rank Correlation*, TR-1314, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, Maryland (August 1983).

[Horowitz 74] S.L. Horowitz and T. Pavlidis, "Picture Segmentation by a Directed Split-and-Merge Procedure," *Proc. 2nd Int. Jnt. Conf. on Pattern Recognition*, Copenhagen, Denmark, pp. 424–433 (13–15 August 1974).

[Jaynes 83] E.T. Jaynes, "Where Do We Stand on Maximum Entropy," in R.D. Rosenkrantz (ed.), *Papers on Probability, Statistics, and Statistical Physics*, D. Reidel Publishing Co. (1983).

[Kender 76] J.R. Kender, *Saturation, Hue, and Normalized Color: Calculation, Digitization Effects, and Use*, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania (November 1976).

[Kender 77] J.R. Kender, "Instabilities in Color Transformations," *IEEE Conf. on Pattern Recognition and Image Processing*, Troy, New York, pp. 266–274 (6–8 June 1977).

[Laws 80] K.I. Laws, *Textured Image Segmentation*, Ph.D. Thesis, Report USCIPI 940, USC Image Processing Inst., University of Southern California, Los Angeles, California (January 1980).

[Laws 82] K.I. Laws, *The PHOENIX Image Segmentation System: Description and Evaluation*, Technical Note 289, Artificial Intelligence Center, SRI International, Menlo Park, California (December 1982).

[Nagin 77] P.A. Nagin, A.R. Hanson, and E.M. Riseman, *Region Extraction and Description through Planning*, Computer and Information Science Report 77-8, University of Massachussets at Amherst (May 1977).

[Nagin 78] P.A. Nagin, *Segmentation Using Spatial Context and Feature Space Cluster Labels*, Computer and Information Science Report 78-8, University of Massachussets at Amherst (May 1978).

[Nevatia 82] R. Nevatia, *Machine Perception*, Prentice-Hall, Englewood Cliffs, New Jersey (1982).

[Ohlander 75] R. Ohlander, *Analysis of Natural Scenes*, PhD. Thesis, Dept. of Computer Science, Carnegie-Mellon University, Pittsburg, Pennsylvania (April 1975).

[Ohlander 78] R. Ohlander, K. Price, and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," *Computer Graphics and Image Processing*, Vol. 8, No. 3, pp. 313-333 (December 1978).

[Ohta 80a] Y. Ohta, T. Kanade, and T. Sakai, "Color Information for Region Segmentation," *Computer Graphics and Image Processing*, Vol. 13, pp. 222-241 (1980).

[Ohta 80b] Y. Ohta, *A Region-Oriented Image-Analysis System by Computer*, Ph.D. Thesis, Dept. of Information Sciences, Kyoto University, Japan (March 1980).

[Pietikäinen 82] M. Pietikäinen, *Image Texture Analysis and Segmentation*, Ph.D. Thesis, Department of Electrical Engineering, University of Oulu, Oulu, Finland, Acta Universitatis Ouluensis, Series C, Technica No. 21, Electronica No. 1 (February 1982).

[Postaire 81] J.G. Postaire and C.P.A. Vasseur, "An Approximate Solution to Normal Mixture Identification with Application to Unsupervised Pattern Classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 2, pp. 163-179 (March 1981).

[Price 76] K.E. Price, *Change Detection and Analysis in Multi-Spectral Images*, Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania (December 1976).

[Price 79] K.E. Price, "Segmentation," *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Chicago, Illinois, pp. 512-514 (6-8 August 1979).

[Rauch 84] H.E. Rauch, "Probability Concepts for an Expert System used for Data Fusion," *The AI Magazine*, Vol. 5, No. 3, pp. 55–60 (Fall 1984).

[Robertson 73] T.V. Robertson, P.H. Swain, and K.S. Fu, *Multispectral Image Partitioning*, TR-EE 73-26, LARS Information Note 071373, School of Electrical Engineering, Purdue University, W. Lafayette, Indiana (August 1973).

[Rosenfeld 76] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-6, No. 6, pp. 420–433 (June 1976).

[Salton 83] G. Salton, E.A. Fox, and H. Wu, "Extended Boolean Information Retrieval," *Communications of the ACM*, Vol. 26, No. 12, pp. 1022–1036 (December 1983).

[Selfridge 82] P.G. Selfridge, *Reasoning about Success and Failure in Aerial Image Understanding*, Ph.D. Thesis, TR103, Computer Science Dept., University of Rochester, New York (May 1982).

[Shafer 82] S. Shafer and T. Kanade, "Recursive Region Segmentation by Analysis of Histograms," *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Paris, France, pp. 1166–1171 (May 1982).

[Sloan 75] K.R. Sloan and R. Bajcsy, "A Computational Structure for Color Perception," *Proc. ACM '75*, Minneapolis, Minnesota (1975).

[Tenenbaum 74a] J.M. Tenenbaum, T.D. Garvey, S. Weyl, and H.C. Wolf, *An Interactive Facility for Scene Analysis Research*, Technical Note 87, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (January 1974).

[Tenenbaum 74b] J.M. Tenenbaum, T.D. Garvey, S.A. Weyl, and H.C. Wolf, *ISIS: An Interactive Facility for Scene Analysis*, Technical Note 95, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California (June 1974). Also published in *Proc. 2nd Int. Jnt. Conf. on Pattern Recognition*, pp. 123–125, Copenhagen, Denmark (13–15 August 1974).

[Tomita 73] F. Tomita, M. Yachida, and S. Tsuji, "Detection of Homogeneous Regions by Structural Analysis," *Proc. 3rd Int. Jnt. Conf. on Artificial Intelligence*, Stanford, California, pp. 564–571 (20–23 August 1973).

[Tsuji 73] S. Tsuji and F. Tomita, "A Structural Analyzer for a Class of Textures," *Computer Graphics and Image Processing*, Vol. 2, No. 3/4, pp. 216–231 (December 1973).

[Wolfe 70] J.H. Wolfe, "Pattern Clustering by Multivariate Mixture Analysis," *Behavioral Research*, Vol. 5, pp. 329–350 (1970).

APPENDIX I

Image-to-Image Correspondence: Linear-Structure Matching
By: Grahame B. Smith and Helen C. Wolf

IMAGE-TO-IMAGE CORRESPONDENCE: LINEAR-STRUCTURE MATCHING

By: Grahame B. Smith, Senior Computer Scientist
Helen C. Wolf, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

Abstract

We examine the task of matching images of a scene when they are taken from very different vantage points, when there is considerable scale change, and when the image orientations are unknown. We use the linear structures in the scene as the basis of our correspondence procedure. This paper considers the problem of describing the linear structures in a manner that is invariant relative to the variations that can occur among images, and discusses a method of finding the best description of the linear structures.

1. Introduction

When the human visual system is presented with two views of a single scene, it determines the relative viewing positions of the two images and brings the latter into correspondence. That is, the relationship of each image to the scene is understood so that both images can be used as information sources for further processing. This human ability functions well over a wide range of viewing positions and conditions. It is this ability to place two very different views of a single scene into correspondence that we address in this paper.

We should draw a distinction between two forms of the image correspondence task. Traditionally, image registration has been a task undertaken by photogrammetrists. One application involves registering an image to a map so that new information, present in the image, may be transferred to the map. Another is the registration of the two images of a stereo pair so that disparity information can be extracted. In each of these tasks the two images, (or, in the first instance, the image and the map), are similar in terms of both their viewing position and their scale. The techniques used for registering the two images are point-based. A feature point in one image is matched to the same feature point in the other image. In automated systems this is achieved by selecting a small window about the feature in one image and then correlating this window with one in the second image. If there is little distortion or occlusion, this technique performs well; it has become the basis of current automated image-registration systems.

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract MDA903-83-C-0027 and by the National Aeronautics and Space Administration under Contract NASA 9-16664. These contracts are monitored by the U.S. Army Engineer Topographic Laboratory and by the Texas A&M Research Foundation for the Lyndon B. Johnson Space Center.

The other form of the image correspondence task seeks to find the relationship among views that differ widely in vantage point, scale, etc. We will refer to this as the correspondence task, and use registration as the name for the form of the task outlined above. In correspondence tasks there is significant distortion between the images, the scale may differ and may not even be constant across a single image, as is the case in oblique aerial imagery, occlusion is common, and the response of the various sensors to a single feature differs greatly. Feature point matching, as used in image registration, is prone to error. However, feature point matching is not the only means of placing images into correspondence. It appears that the human visual system makes use of nonpoint features, such as linear structures and extended landmarks. The aspects of our investigation reported here utilize the linear structures of the images as the prime elements for achieving correspondence.

In classifying the methods that could be employed to find linear structures in images, we draw a distinction between techniques that use semantic information and those that do not. If, for example, we apply a road operator to locate some of the linear structures in an image, that operator has had built into it semantic knowledge about the appearance of roads. We could proceed in this manner and build comparable operators for all the scene objects that manifest themselves as linear structures in images. Alternatively, we could seek to find the linear structures in an image without "identifying" their nature. In this case, we identify the image behaviour interpreted by us as a linear structure without knowledge of the world objects that gave rise to that structure. We choose this latter course because we wish to establish the correspondence among images without first having to identify the scene objects.

The correspondence task is carried out in three stages: we must find the linear structures, we must build their descriptions and, finally, we must match these descriptions. The details of the first stage is reported in Fischler and Wolf [1]. In this paper we explain how those procedures are employed in the correspondence task. We present a detailed account of our implementation of the second stage – namely, building structure descriptions – along with an outline showing how these descriptions are to be used in the final matching stage.

2. Finding the Linear Structures

Descriptions of the semantically free procedures we use to find linear structures in images can be found in Fischler and Wolf[1]. In essence, these procedures first find those pixels whose intensity levels are local maximums and minimums, then cluster such pixels and identify the minimal spanning tree for each cluster. The long paths in each of the spanning trees are found, whereupon these form the basis for the linear structure reported by the procedures. The results of applying these procedures are shown in Figures 1-4. Figure 1 is a natural-color oblique view of the Eel river in northern California; Figure 2 is a vertical infrared view of the same scene. Each was scanned through red, green, and blue filters; the results of the procedures for finding linear structures in each of these separation images are shown in Figures 3(a),3(c),3(e) and 4(a),4(c),4(e). In addition, the red, green, and blue separation images were combined into images of hue, saturation, and intensity; these were also processed to find the linear structures contained in them. The results are shown in Figures 3(b),3(d),3(f) and 4(b),4(d),4(f).

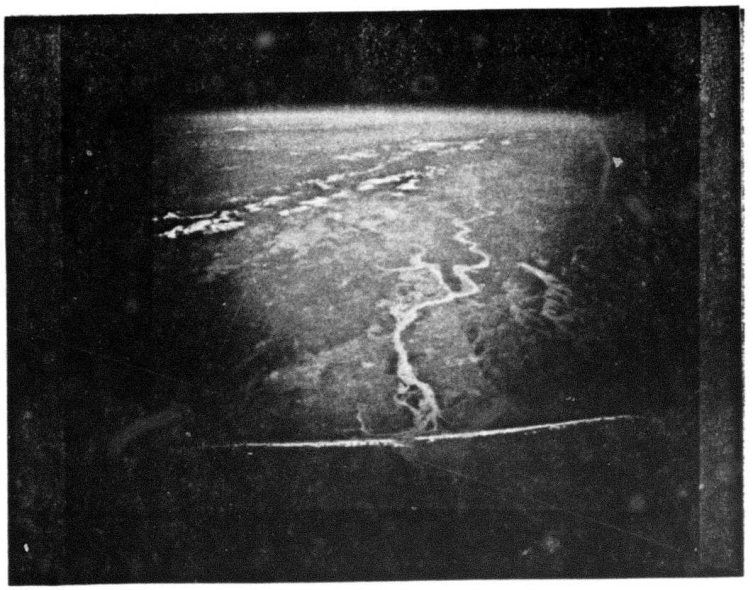


Figure 1. Oblique Natural-Color Image of the Eel River

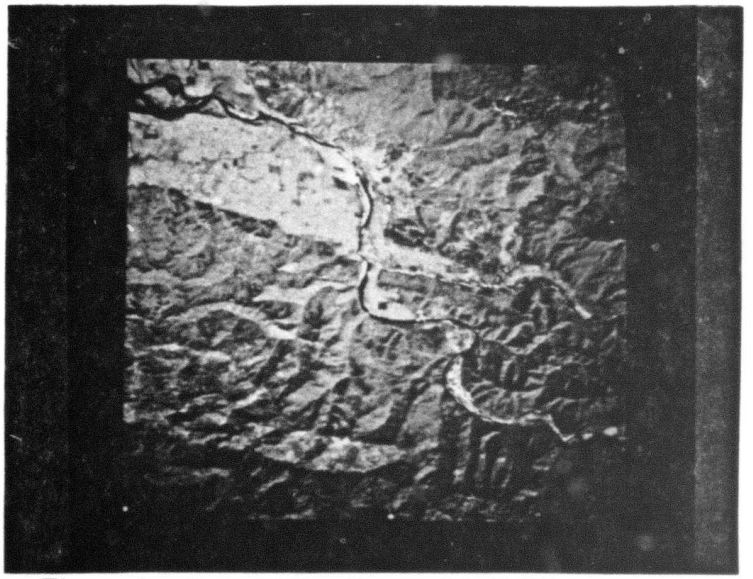


Figure 2. Vertical Infrared Image of the Eel River

These separation images differ appreciably in their linear structure. Certainly no one separation image can be selected as providing a complete delineation of the river. The philosophy we adopt is to view the original image from as many

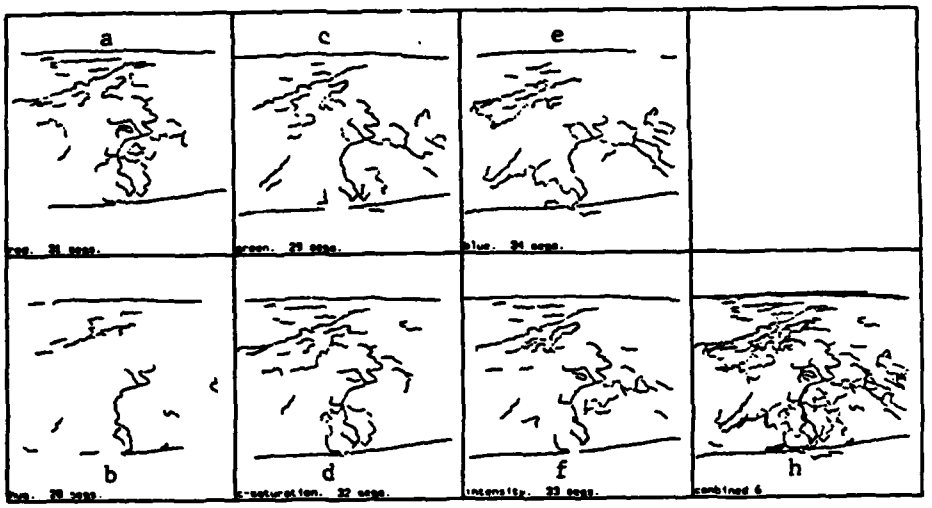


Figure 3. Linear Structure in the Oblique Image

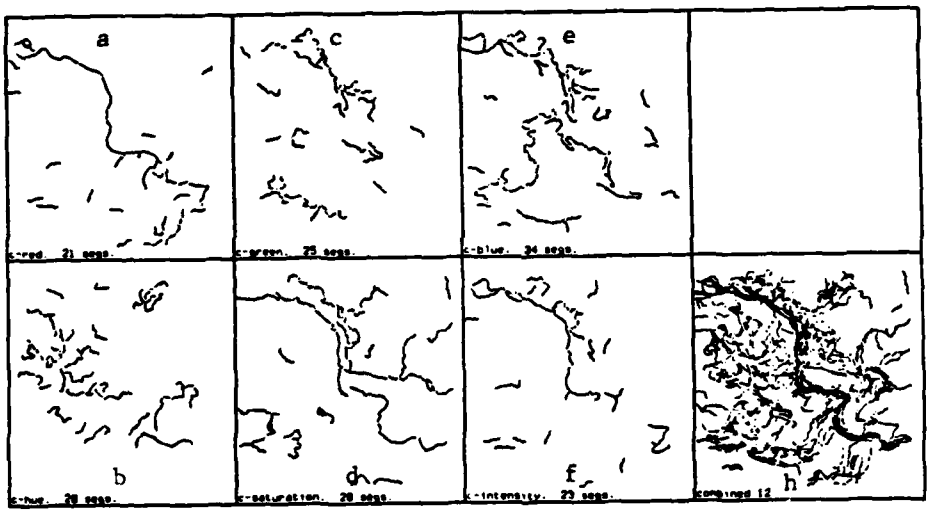


Figure 4. Linear Structure in the Vertical Image

perspectives as possible, obtaining the linear structures as seen from each of these. That is, we look for structures in hue, in the green spectral band, and so on. Of course, the hue image is derived from the red, green, and blue images, and contains only redundant information, but this presentation of the information may show structure that was masked in other presentations. In this sense, the additional

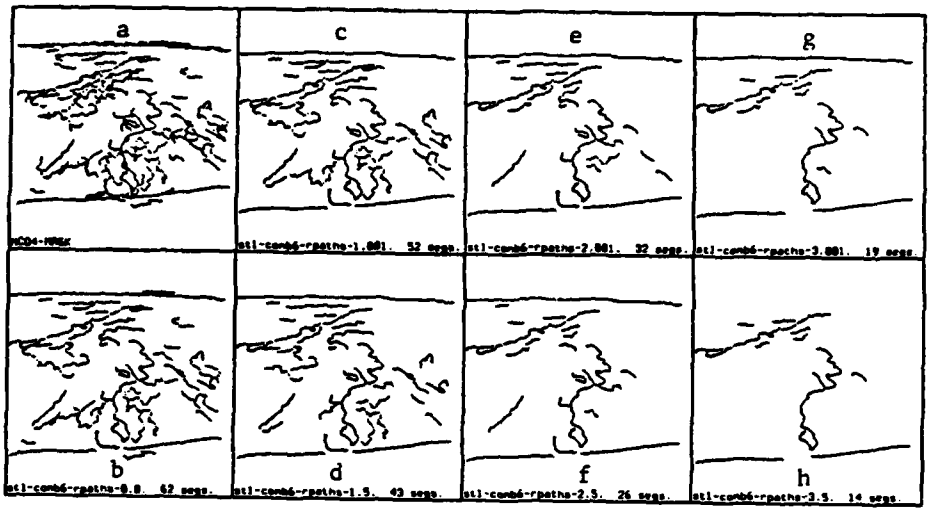


Figure 5. Linear Structure in the Composite Oblique Image

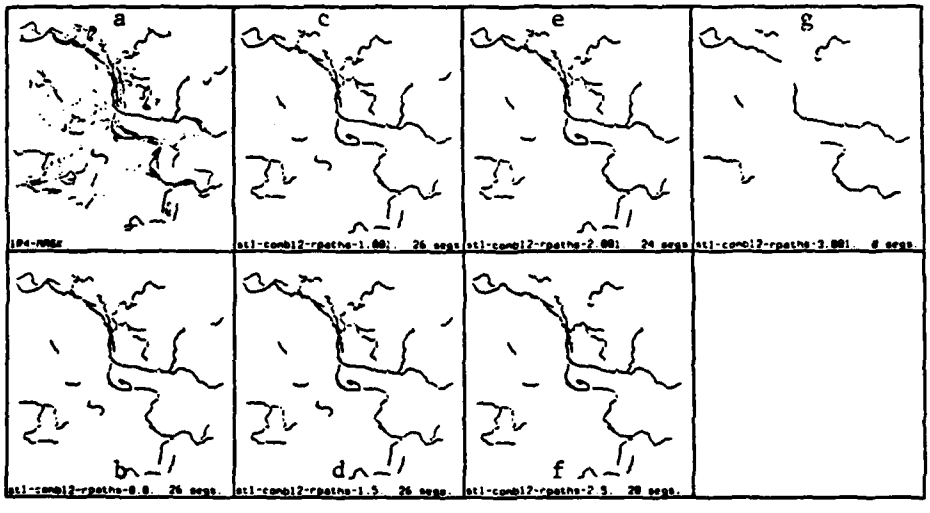


Figure 6. Linear Structure in the Composite Vertical Image

perspectives provide new information on which the linear-structure finders can act. The results of combining the linear structures extracted in all the various perspectives are shown in Figures 3(h) and 4(h). Clearly, some of this structure comes from shading effects rather than from physical structure in the scene. We need to separate the real physical structure from all else.

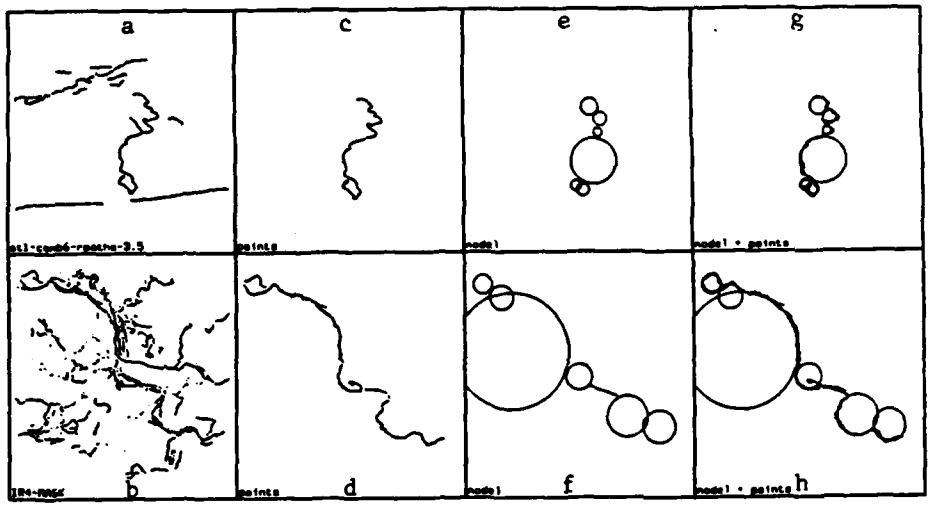


Figure 7. Structure Descriptions

Figures 3(h) and 4(h) were obtained by adding the binary images produced by the linear-structure finders. Consequently, in the combined image the values are greater than one at those pixel positions where linear structure was seen in more than one separation image. We treat this combined produce as a new "grey-level" image and, once again, apply the linear-structure finders. The results obtained from applying these procedures to Figures 3(h) and 4(h) are depicted in Figures 5(b) and 6(b). Figures 5(a) and 6(a) show an intermediate result before we cull short structures. For each of the structures in Figures 5(b) and 6(b), we calculate the average "intensity", that is the average number of original separation images exhibiting that linear structure. Figures 5(c),5(d),5(e),5(f),5(g),5(h) and 6(c),6(d),6(e),6(f),6(g),6(h) reveal which segments would remain if we thresholded the "intensity" values at 1, 1.5, 2, 2.5, 3, and 3.5, respectively.

We build a description of the linear structures from one of these images. The image we use will depend on the final matching procedure. If we wish to attempt

to first match the "strongest" structures we use the image resulting from a high threshold. On the other hand, if we wish to match the complete structure, the unthresholded image would appear to be more appropriate. In the next section, where we discuss the nature of the structure description, we use as examples the foregoing two extremes. In the case of the oblique image, we have used the "intensity" image at a threshold of 3.5 (Figure 7a), while for the other case, the vertical infrared image, we employ the unthresholded image (Figure 7b).

3. Describing the Linear Structures

The means used to describe a linear structure is not independent of the use to which this description will be put. A description that makes it possible to reproduce the structure is different from one that is sufficient to recognize it. As matching is our goal, we want a description that is general enough to be unaffected by noise in the data, but specific enough to distinguish among structures that the human visual system would classify as different. To the extent feasible, the description must be invariant with respect to the variations that can occur in the data. Specifically, we want the description to be independent of orientation, scale, and vantage point.

Our matching process will compare graphs of symbolic descriptions. We will use as little metric information as possible. Consequently, the descriptions we employ are symbolic ones, the primitive entities in each of which have qualities that are themselves symbolic. For example, a primitive may be a straight-line segment whose properties, such as an intersection angle (with some other primitive), have values *acute*, *near-colinear*, etc. rather than a value in degrees.

The primitives we have chosen to use are straight-line segments, arcs of circles, and model-less, that is, data we prefer to describe as indescribable, data for which the data set itself is the most apt description. The choice of these few primitives stems from the observation that human description of linear structures seems to be based on curves and straight lines – moreover on whether adjoining curves curve the same or opposite ways and whether adjoining pieces of the structure intersect in particular ways. It is also a fact that humans find certain parts of the structure too difficult to describe, and assign them some generic term like "wiggles".

Selection of the description primitives is only half the task of description building. We need to be able to divide the linear structure into parts and assign a primitive to each. Usually the task of dividing the linear structure into parts and describing each of these parts has been handled as two relatively independent processes in which partitioning has preceded parts description. The difficulty with this approach is that some characterization of the breakpoints between parts has to be found. Generally, this characterization is based only on local properties of the linear structure, even though neighborhood information or local inhibition may be employed so as to benefit from more broadly based information. In this respect, the task of describing a structure in terms of its primitive parts appears to have been replaced by the more difficult undertaking of describing breakpoints. Our concern is to find the "best" description without first having to find the "best" subdivision. Furthermore, we would like "best" to be defined in terms of a global criterion rather than local properties of the structure.

The advantage of defining best in terms of a local criterion is that many candidates for the definition of "best" spring to mind. The disadvantage of defining

"best" in a global sense is the lack not only of likely definitions, but also of computationally effective algorithms for finding this optimal solution. However, a description that views the data from a "gestalt" perspective seems more likely to be independent of image orientation, scale, and vantage point than one that applies local data measures to define the optimal description. We define best description, as the one that minimizes the number of symbols needed to encode the linear structure in terms of our description primitives.

4. Minimal Encoding

The need to match data to description primitives is a central aspect of decision theory and pervades artificial intelligence research. It is a human's ability to abstract data in terms of descriptive models that distinguishes human information processing from its electronic namesake. Effective data abstraction is a balance between two competing requirements. On the one hand a descriptive model must fit the data adequately, while, on the other, the descriptive model must not be needlessly complex. The criterion we use to select among competing descriptions is based on the work of Georgeff and Wallace [2], in which the description considered "best" is the one that can be encoded in the fewest symbols.

Suppose we wish to send data to some receiver so that he can recreate the data to some preselected level of resolution. The sender and receiver have agreed on a language for this communication that consists of a set of primitive elements. What is the most efficient encoding of the data; which message has the minimal encoding length? Consider the example of sending a message that describes a linear

structure. The latter can be thought of as a list of x and y coordinates. Let us further suppose that the language of communication contains three primitives: straight-line-segments, arcs-of-circles, and model-less-segments. Is it more efficient to send the data as a single model-less-segment primitive, that is, as a list of (x,y) coordinates, or might it be more efficient to describe the data by one or more of the other primitives, specifying sufficient information to describe how the actual data differ from the primitives?

The message can be viewed as a list

$$((M_1, D_1), (M_2, D_2), ...) \quad ,$$

where M is the specification of the primitive, D the specification of the data in terms of the selected primitive M. Let us consider an example. Suppose we have a data set that approximates a straight-line segment. We could communicate this by specifying a straight-line-segment primitive M, where M consists of a code for the straight-line-segment primitive and parameters that specify the actual straight line segment. These parameters might be the endpoints of the line. We also need to specify the actual data in terms of this primitive M. The data specification D might, for each data point, specify its coordinates as a distance along the line (from its centre) and the perpendicular distance from the point to the line.

As the expected distances from the points to the line are small, we shall choose an encoding of these distances so that the more probable of these, the smaller distances, are encoded in fewer symbols (or bits) than those that are less likely. In the actual examples we shall describe later, we assumed a Gaussian distribution for these perpendicular distances and we encoded optimally for that distribution. The

optimal encoding length is just the negative logarithm of the probability, i.e., the function denoted as "information" in information theory.

If we have a small number of data points fewer symbols may be needed to communicate the data as a list of points; if, however, there is a large number of data points that exhibit behaviour consistent with a primitive, it will probably be cheaper to encode this data set as the primitive and then specify the data in terms of that primitive. Of course we are not just comparing the encoding of all the data with either one primitive or another. It might be more efficient to encode the data as a few primitives, with each primitive "explaining" a different part of the data. The encoding we select is the one that is globally best in explaining all the data.

A way of viewing the message form outlined above,

$$((M_1, D_1), (M_2, D_2), ...) \quad ,$$

is to look upon M as the overhead of introducing another primitive while D represents the quality of the fit between the data and the primitive. Of course, since different primitives have different M's, M also weights each primitive's efficiency for encoding data. In comparing message length we are balancing the complexity introduced by adding an extra primitive to the description of the data against the quality of fit between the assembled primitives and the data values.

Although the above discussion focused on encoding messages for communication, we use minimal encoding length as the criterion for finding the best description of a linear structure - without any interest on our part in actually transmitting the data. This of course means that we only have to decide how many symbols would be used if we were to encode the linear structure in a particular manner rather

than actually doing the encoding. We can use the results of information theory to determine the optimal encoding length without even having to understand what the optimal encoding scheme is. That is, information theory gives us an operator, or a measure, that we can apply to a description to determine how many symbols we would need if we were to encode it optimally, without any consideration of the actual encoding scheme and without the need to do the encoding.

Let us consider our application, encoding linear structures in terms of three primitives: straight-line-segments, arcs-of-circles, amd model-less-segments. We will assume that the data are specified on a NxM grid, and that the noise in the data will induce a Gaussian distribution of the data points around the generating primitive. Given that all grid points are equally likely, the cost in bits of encoding a grid point is $logN + logM$, (log is to the base 2). Now consider the three alternative ways of encoding r data points (using one primitive only).

Model-less-segment:

We need a code to specify that the primitive being used is the model-less-segment. As there are only three primitives, and we assume that they are all equally likely, it costs $log3$ bits to specify the code. Specification of the data in terms of this primitive will require in turn that we specify r grid coordinates, that is, a cost of $r(logN + logM)$ bits.

Straight-line-segment:

We can specify the straight-line-segment primitive by specifying the endpoints of the line segment. This costs $2(logN + logM)$ bits. In addition, the cost of specifying the code for this primitive is $log3$. To specify the data in terms of this primitive we will, for each data point, specify a distance along the line and

the perpendicular distance from the point to the line. If the line segment is of length l (in grid units) then, to specify r distances, if we assume that all distances are equally likely, will cost $rlogl$ bits. If it is also assumed that the data points have a Gaussian distribution about the primitive model, the cost of specifying r perpendicular distances is

$$\sum_{r\,pts} -log\left(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-d^2}{2\sigma^2}}\right) \quad ,$$

where d is the perpendicular distance from the point to the line, and σ the standard deviation associated with the distribution. When the above expression is expanded, the sum over the d's is just the sum of the residuals squared that is calculated when the line is fitted to the data by least-squares methods.

Arcs-of-circles:

We specify the arcs-of-circles primitive by specifying the endpoints of the arc and one other point on the arc. This costs $3(logN + logM)$ bits, while the cost of specifying the code for this primitive is $log3$ bits. To specify the data we use the same scheme as we did for the straight-line-segment primitive.

Using these costing functions and a search algorithm that examines the various ways for partitioning a linear structure into primitives, we find the best description of that structure.

5. Results

The results of using the foregoing procedure on some of the linear segments found in Figures 1 and 2, (and shown in Figures 7(a) and 7(b)), are depicted in the remaining panes of Figure 7. From Figures 7(a) and 7(b) we have selected some

linear structures. The selected structures, which form the main course of the Eel river, are shown in Figures 7(c) and 7(d). Our interest is in determining whether the description built from one image is the same as that from the other. Of course, in the final version of the structure builder we would need to handle all the segments simultaneously, but this will necessitate considerable improvement in the search algorithm to keep computational costs down to a reasonable level.

Figures 7(e) and 7(f) show the primitives returned. The arc of circles are shown as full circles to improve readability. In Figures 7(g) and 7(h) the primitives have been overlaid on the data to show the quality of fit. In assessing these results, one should keep the purpose of this description in mind. We want to extract a description of the linear structure in terms of lines and curves, in terms of the manner in which parts intersect (acute angles, near-colinearity, etc.), in terms of relative curvature (tight curves, gentle curves, and the like), and in terms of the sequencing of parts in the structure. Given that the two images are viewed from very different vantage points, that the scale is quite different (not even constant in one image), that one image was taken in the infrared band and one in the visible band, that the images were taken one-and-a-half years apart during different seasons, and that no semantic information was used in the processing, the closeness of the resulting descriptions is noteworthy. This points to the usefulness of processing the data in the above manner; namely, the method of finding the linear structures; the primitives used to encode the structure; and the encoding-length measure as a criterion for best description.

Figure 7 shows the results obtained with real data. Similar results have been obtained in experiments that employ other real data sets. Justification of the

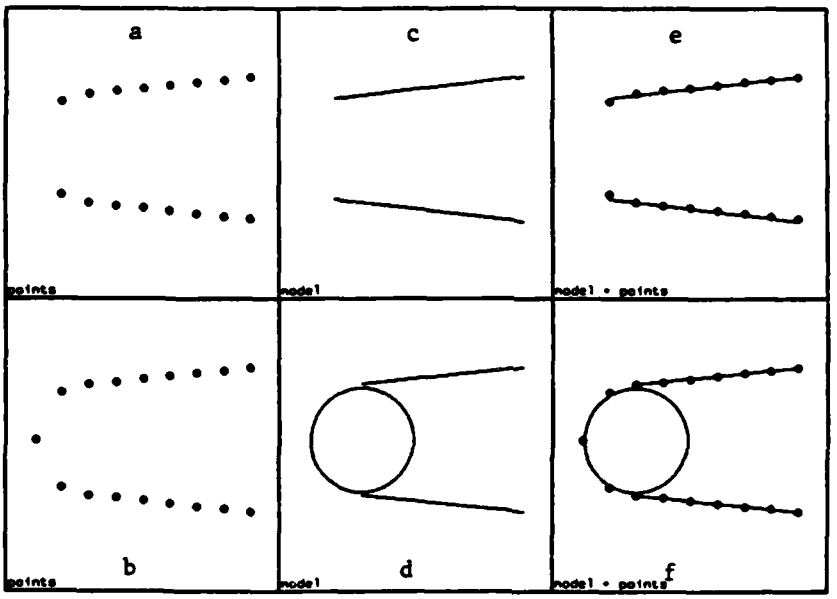


Figure 8. Encoding of Synthetic Data

method, however, requires further extensive experimentation. To better understand the behavior of the description builder we include an example using synthetic data. The data points are shown in Figures 8(a) and 8(b). In Figure 8(b) one extra data point has been added to those shown in Figure 8(a). The resulting descriptions are shown in Figures 8(c) and 8(d) and overlaid on the data in Figures 8(e) and 8(f). The addition of one critical point alters the description, an effect not unknown in the human visual system. The resulting descriptions seem to match those perceived by humans when they are presented with Figures 8(a) and 8(b). While we could not claim that minimal encoding is the criterion used by the human visual system for description building, we note that this criterion conforms to the type of behavior we would want to achieve if we were modeling the visual system. Of course, if the resultant description is sensitive to every addition or deletion of a data point it is of

little use. In general, the minimal-encoding-length description appears to be stable with respect to data changes, except when "critical" points are added or deleted.

6. Matching the Descriptions

If the description we obtain from the description builder characterizes the data and is invariant with respect to orientation, scale, and vantage point, the burden of matching descriptions is lightened considerably. It is our intent to match descriptions at the symbolic level, to represent the descriptions found by minimal encoding as graphs of symbolic entities, and to match those graphs on the basis of their structure. Of course, it is unlikely that the graphs derived from different images will match perfectly. Nevertheless, from a prospective match we can find *correspondences in the original images*, and calculate the camera transformation between the images.

This procedure allows data in one image to be transformed into the other. It means that we can transform a linear structure found in one image into the other image. For those parts of the graph where there is a mismatch we can ask the question: how would the linear structure that is associated with the mismatch be encoded if it were first transformed into the other image and then encoded? In this manner we can attempt to explain the graph mismatches. If we cannot explain the mismatches we should consider another match of the graphs. Through this process of hypothesis and verification, we seek to avoid acceptance of a transformation that does not explain "all" the data.

7. Conclusion

Having found the linear structures in an image, we are faced with two major tasks before we can use these structures to find the correspondence between different images of a scene. We need to be able to describe these structures in a way that is independent of the variations that can occur between the images, and we need to be able to match these descriptions to find the relationship between the images.

In considering structure description we show that the usual technique of dividing the structure into parts and then describing the latter can be replaced by a procedure that finds the "best" description of the data on the basis of a global view of that data. This technique simultaneously divides the structure into parts and describes them. "Best" is defined as the cheapest encoding of the data when we consider the trade-off between the quality of explanation of the data and the complexity of that explanation.

This approach produces a description of linear structures that appears relatively insensitive to the vantage point, scale, and orientation of the original images. It may prove to be a description that enables easy matching, and hence an effective approach to solving the problem of image-to-image correspondence.

References

1. Fischler, M.A. and Wolf, H.C., Linear Delineation, *Proceedings of Computer Vision and Pattern Recognition Conference*, Washington, D.C. 1983, pp 351-356.

2. Georgeff, M.P. and Wallace, C.S., A General Selection Criterion for Inductive

Inference, in: O'Shea, T. (Ed.), *Proceedings of Advances in Artificial Intelligence, Pisa, Italy, September* 1984, North-Holland, Amsterdam, 1984.